ENGINEERING

LEVEL II

12

USCIPI Report 1000

UNIVERSITY OF SOUTHERN CALIFORNIA

COMPUTATIONAL MODELS FOR TEXTURE ANALYSIS AND TEXTURE SYNTHESIS

by

David Donovan Garber

May 1981

Department of Electrical Engineering
Image Processing Institute
University of Southern California
Los Angeles, California 90007

Research sponsored by
Defense Advanced Research Projects Agency
Contract No. F-33615-80-C-1080
DARPA Order No. 3119

IPI

IMAGE PROCESSING INSTITUTE

(12) USCIPI Report 1000

COMPUTATIONAL MODELS FOR TEXTURE ANALYSIS AND TEXTURE SYNTHESIS

by

David Donovan Garber

May 1981

Department of Electrical Engineering

Image Processing Institute

University of Southern California

Los Angeles, California 90007

Research sponsored by:

Defense Advanced Research Projects Agency

Contract No. F-33615-80-C-1080

DARPA Order No. 3119

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) USCIPI-1000 ✓	2. GOVT ACCESSION NO. AD-A102 470	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) COMPUTATIONAL MODELS FOR TEXTURE ANALYSIS AND SYNTHESIS.		5. TYPE OF REPORT & PERIOD COVERED (9) Technical Report, May 1981
		6. PERFORMING ORG. REPORT NUMBER (14) USCIPI Report 1000
7. AUTHOR(s) (10) DAVID DONOVAN GARBER		8. CONTRACT OR GRANT NUMBER(s) (15) F-33615-80-C-1080
9. PERFORMING ORGANIZATION NAME AND ADDRESS Image Processing Institute University of Southern California University Park, Los Angeles, Ca. 90007		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DARPA Order No. 3119
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE (11) May 1981
		13. NUMBER OF PAGES 254 pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Air Force Wright Aeronautical Laboratories Wright Patterson AFB, Ohio 45433 (12) 257		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for release: distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image processing, texture synthesis, digital image texture, natural texture, texture segmentation, texture identification, texture analysis, image segmentation, texture simulation, stochastic processes, image understanding.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Numerous computational methods for generating and simulating binary and grey-level natural digital-image textures are proposed using a variety of stochastic models. Pictorial results of each method are given and various aspects of each approach are discussed. The quality of the natural texture simulations depends on the computation time for data collection, computation time for generation, and storage used in each process. In most cases, as computation time and data storage increase, the visual match between the texture simulation and the parent texture improves. Many textures are adequately

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

391 141

simulated using simple models thus providing a potentially great information compression for many applications.
In most of the texture synthesis methods presented in this thesis, pixel values are generated one-at-a-time according to both the given model and the values of pixels previously generated in the synthesis until the image space is completely filled. Nth-order joint density functions estimated from a natural texture sample were used for this purpose in one method. The results are excellent but the storage required, even for binary textures, is large. Therefore, a much simpler first-order linear, autoregressive model was applied to the texture synthesis problem. Using this model on both binary and continuous-tone textures, each pixel is generated as a linear combination of previously generated pixels plus stationary noise. The results indicate that many textures are satisfactorily simulated using this approach.

By adding cross-product terms, the first-order linear model is extended to a second-order linear model. The simulation results improve slightly but the number of computations required for the statistics collection process increases drastically. Non-stationary noise was then used in the synthesis process and further improvements in the quality of the simulations are achieved at the cost of increased storage.

Methods of texture simulation using more than one model are studied in this thesis. These multiple-model are useful for many textures, especially those with macro-structure. They also improve the fit of the model when applied to the parent texture data and therefore may produce improved simulations.

A final model, called the best-fit model, generates texture simulations directly from the parent texture itself. Each pixel in the synthesis image is generated based on the similarity of its previously-generated, neighboring pixel values to pixel values in all similarly-shaped neighborhoods in the parent texture. The measures of similarity at all points in the parent texture, along with a random variable, are used to generate the next pixel value in the synthesized image. The synthesis results using model are excellent but the synthesis process is very computationally demanding.

Although the success of texture synthesis is highly dependent on the texture itself and the modeling method chosen, general conclusions regarding the performance of various techniques are given. Methods of texture segmentation and identification based on texture synthesis results are also presented.

ACKNOWLEDGEMENTS

I would like to thank the chairman of my thesis committee, Dr. Alexander A. Sawchuk, who enabled me to do this research under my own initiative while providing painstaking direction and support during my thesis work. His guidance and noteworthy efforts made this possible.

I would also like to thank the following people for their individual contributions to this work: Drs. Tim Strand and Alan Schumitzky for their helpful suggestions as members of my dissertation committee, Dr. Harry Andrews for providing financial support and direction as my guidance committee chairman during the first years of my work at USC, Dr. O.D. Faugeras for his assistance on Chapter 4, Ray Schmidt for his fine photographic work, Gary, Tom, Chuck and Bret for undertaking an assortment of necessary programming and monkey work over the years, Dr. Keith Price (IPI's computer wizard) and Paul Liles for programming support, and Dr. Frederic Carlson for agreeing to an arrangement whereby I was able to consume hundreds of thousands of dollars of computer time at reasonable cost.

Special thanks belong to Hilda (zlfrpergubarl) for her excellent preparation and organization of the final work.

My years at USC have been truly enjoyable. I repeatedly extended my stay here as my life became more comfortable day by day and my experiences have been both satisfying and rewarding. The atmosphere which has been created by my interaction with the people around me has tended to make me quite complacent. I believe that it is good to stop along the road and smell the flowers.

Finally, I will always be grateful to my family for their love and support through the years.

Life is wonderful.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Numerous computational methods for generating and simulating binary and grey-level natural digital-image textures are proposed using a variety of stochastic models. Pictorial results of each method are given and various aspects of each approach are discussed. The quality of the natural texture simulations depends on the computation time for data collection, computation time for generation, and storage used in each process. In most cases, as computation time and data storage increase, the visual match between the texture simulation and the parent texture improves. Many textures are adequately simulated using simple models thus providing a potentially great information compression for many applications.

In most of the texture synthesis methods presented in this thesis, pixel values are generated one-at-a-time according to both the given model and the values of pixels previously generated in the synthesis until the image space is completely filled. Nth-order joint density functions estimated from a natural texture sample were used for this purpose in one method. The results are excellent but the storage required, even for binary textures, is large. Therefore, a much simpler first-order linear, autoregressive model was applied to the texture synthesis problem. Using this model on both binary and continuous-tone textures, each pixel is generated as a linear combination of previously generated pixels plus stationary noise. The results indicate that many textures are satisfactorily simulated using this approach.

By adding cross-product terms, the first-order linear model is extended to a second-order linear model. The simulation results improve slightly but the number of computations required for the statistics collection process increases drastically. Non-stationary noise was then used in the synthesis process and further improvements in the quality of the simulations are achieved at the cost of increased storage.

Methods of texture simulation using more than one model are studied in this thesis. These multiple-model are useful for many textures, especially those with macro-structure. They also improve the fit of the model when applied to the parent texture data and therefore may produce improved simulations.

A final model, called the best-fit model, generates texture simulations directly from the parent texture itsel Each pixel in the synthesis image is generated based on the similarity of its previously-generated, neighboring pixel values to pixel values in all similarly-shaped neighborhoods in the parent texture. The measures of similarity at all points in the parent texture, along with a random variable, are used to generate the next pixel value in the synthesized image. The synthesis results using model are excellent but the synthesis process is very computationally demanding.

Although the success of texture synthesis is highly dependent on the texture itself and the modeling method chosen, general conclusions regarding the performance of various techniques are given. Methods of texture segmentation and identification based on texture synthesis results are also presented.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Texture is important characteristic for the analysis of many types of images. It is an important feature for discrimination and identification of regions in images and as a result the vast majority of work on texture has concentrated on these applications. Although many different texture discrimination techniques have been developed, most are *ad hoc*.

The problem inverse to texture analysis is texture synthesis, or the generation of image fields having analytical and visual characteristics similar to natural textures. Texture synthesis has been over-shadowed by the emphasis placed on the discrimination problem and its applications. Little work has been done on synthesis even though numerous applications exist. For example, intelligent image sensors could transmit boundaries of textured image regions. Based on statistics gathered by the sensor, this region could be reconstructed using simulation techniques with little or no loss of information. The result is excellent data compression.

Texture synthesis can also be used as a texture analysis tool leading to a better understanding of textures and their perception by humans as well as improved methods of discrimination. By carefully controlling the statistics of a texture in a synthesis process visual changes in texture are observed. Thus, texture synthesis methods allow researchers to identify and measure the information content of individual statistical measurements. By assembling these measurements and incorporating them into a texture simulation process, statistics may be measured from a parent texture and used to produce a texture simulation. The degree to which the parent and simulation are visually similar indicates the value of the statistical measurements and the model used in the simulation process. Given a group of statistical measurements which are proposed to be useful texture measures, possibly the best may be chosen based on the quality of the corresponding texture simulations. In this way, researchers are able to develop better discrimination as well as better synthesis methods.

1.2 Concepts of Texture Synthesis

Despite its importance, a precise definition of texture does not exist. Texture is often considered to be composed of a set of primitives and their spatial organization. More important, texture usually possesses an invariance property. We will use this invariance property as one definition of texture in this thesis. An observer should detect no visual difference between one windowed portion of a textured region and another. Thus texture is also a function of window size. If a difference over a region is detected then either the texture is not homogeneous (see section 2.1) or a larger or smaller window should be used. Windowing is very important when gathering statistics to be used for texture discrimination or texture synthesis.

The approach to texture synthesis used in this thesis is outlined in Fig. 1.1. As a first step in the synthesis process, statistics are calculated from measurements taken on a parent sample texture. The statistics are then used to estimate model parameters. In the final step, these model parameter estimates are used to generate a texture synthesis.

All of the digital images in this thesis are 512 by

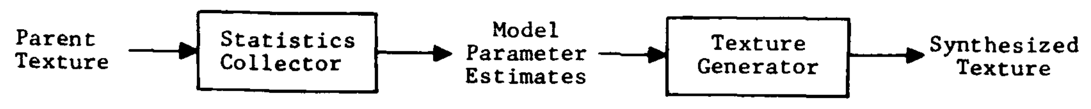


Figure 1.1 Approach to Texture Synthesis

512 pixels. They have either 256 gray levels (continuous tone) or 2 gray levels (binary). The original parent texture images in this thesis have been chosen from an album by Brodatz [1]. High quality prints obtained from the photographer were scanned and digitized at the USC Image Processing Institute.

The performance measure for any texture synthesis method is purely visual. Thus the rating of any synthesis system must be made by an observer and is subject to human variability. These facts must be remembered when considering the results of synthesis techniques discussed in this thesis and in other works. For this reason, the visual analysis of results in this thesis is left primarily to the reader. Nevertheless, general guidelines and trade-offs involved in texture synthesis have been developed in the course of this work.

The success of any texture synthesis as well as any image processing technique (enhancement, restoration, etc.) also depends on the display medium used for final results. It is unfortunate that the product of so much work is subjected to the imperfections of recording and printing processes. We have attempted to minimize these degradations as much as possible.

1.3 The Stochastic Model

The approach used to synthesize textures in this thesis is probabilistic (statistical) rather than structural because the structural approach would probably require a different processing method for each texture. Textures which are irregular and highly random are often difficult to analyze using a structural approach as they do not have "structure" as such. On the other hand, such textures are easily analyzed using a probabilistic approach. Textures which are not composed of well-defined non-varying primitives usually do not lend themselves to a structural approach. Highly regular textures which are usually best explained using a structural approach may also be studied using a statistical approach. In this case, the statistical model must be constructed to explain regular and periodic events with less importance placed on random elements of the model. Naturally, the structural and probabilistic approaches overlap considerably.

In our statistical approach to texture analysis and synthesis we will use various stochastic models. Textures are analyzed as series of pixels to which a model is fit. The time or space series of successive pixels which forms our sample parent texture is regarded as a sample realization from an infinite population of such textures.

A model is derived which attempts to explain the numerical sequence of pixels in the observed parent texture. This stochastic model may then be used to simulate a similar sequence which becomes our texture synthesis.

The power of the stochastic models presented in this thesis is that it is easy to use the model in a mode which synthesizes textures given the necessary model parameters. In this sense, the stochastic approach is sufficient to capture everything about a texture. The quality of the synthesis is also a measure of the amount of information contained in the model.

1.4 Organization and Contributions of the Thesis

In the next chapter, a one-dimensional texture synthesis model is mathematically developed. It is analyzed as a Markov model where the state of the system is considered to be defined by the sequence of previous pixel values. Similar texture synthesis has also been done by Julesz [8], Purks and Richards [3], Conners [7], Abend [6], and Gagalowicz [4,5]. With the simple model presented in Chapter 2, textures with controlled statistical properties are generated and used to study human perception of texture.

In Chapter 3, the one-dimensional model is extended to tw. .imensions and is used to synthesize natural binary textures. This work was presented earlier by Garber [9] and similar work was done by Lu and Fu [10]. This two-dimensional probabilistic model which generates textures based on higher-order probability densities is then reduced carefully to a linear autoregressive model. Results of both methods applied to a wide variety of textures are shown.

In Chapter 4, a method to reconstruct higher-order densities for texture synthesis from second-order measurements is presented. The results confirm the value of second-order statistics in texture analysis.

In Chapter 5, the autoregressive model is applied to continuous-tone texture synthesis. Much simpler synthesis models were used by McCormick and Jayaramamurthy [2] and Tou, Kao and Cheng [11]. The success of their method was not established due to the limited number of textures involved in their studies, however their results suggested that time series models could be used to generate some natural textures. The large, two-dimensional, linear model presented in Chapter 5 is then extended to a quadratic autoregressive form and synthesis results with both stationary and non-stationary noise are presented. The generated textures show that these models define valuable synthesis methods.

In Chapter 6, texture synthesis methods which incorporate more than one autoregressive model are presented. The multiple-model methods methods could be valuable in synthesizing textures which have very coarse structure and textures which are composed of subtextures.

In Chapter 7, a texture synthesis method which simulates complete probability density information for use in the synthesis process is developed. This method is computationally burdensome but also yields the best synthesis results. A method similar to this will be valuable as processor speed increases in the future.

In Chapter 8, methods for adding and removing non-homogeneities in texture mean and variance are presented.

In Chapter 9, the measures used to estimate parameters in the autoregressive model are applied to the problem of texture discrimination. The methods differ from those presented earlier by Deguchi and Morishita [12], Kaiser [13], Pratt, Faugeras and Gagalowicz [14], and Haralick _et al._ [15]. The chapter illustrates two approaches to use statistics from a synthesis model to discriminate textures.

The experimental results of this thesis are largely visual therefore particular attention should be devoted to the figures. A casual reader could read section 2.1, Chapter 3, Chapter 5, and Chapter 7 and still understand much of the basic concepts of the work as well as major results. Chapter 10 contains a final summary and comparison of the models and their corresponding results.

As a whole, this thesis presents results in texture simulation using methods developed herein or only briefly mentioned in other previous studies. The texture syntheses are exceptional in some cases and certainly notable in others. This work should encourage additional research in the field of texture synthesis.

1.5 Notation

Any variables which have different meanings within this thesis are clearly defined in the places where they are used. The term "complex" means complicated rather than a variable with real and imaginary parts. The terms "normal distribution" and "normally distributed" are to be taken in a statistical probability density function (ie.Gaussian distribution) sense.

CHAPTER 2

ONE-DIMENSIONAL BINARY TEXTURE MODEL

2.1 Introduction

Texture is a complex image attribute that has been the subject of much research and is difficult to define precisely. The relationship between discrimination of textures by human observers and the mathematical attributes of textures has also been extensively researched. Models for computer discrimination have been proposed based on statistical parameters considered in some aspects to be primary texture measures.

The terminologies in a portion of previous texture work have often been vague at best. As a result, the terms second-order and third-order have been seriously twisted and misinterpreted from study to study. In this and following chapters, we will attempt to suppress this confusion by carefully defining the various terms.

The stochastic approach toward texture analysis considers texture fields as samples of two-dimensional stochastic fields. Assuming that we are dealing with sampled and quantized imagery, let $I(n_{i1},n_{i2})$ denote the random field. Here n_{i1} and n_{i2} are integers representing coordinates of points in the image plane. Let $\vec{n}_i$ be the vector having coordinates n_{i1} and n_{i2} (i.e.$\vec{n}_i=(n_{i1},n_{i2})$). Second-order statistics are given by the set of second-order joint density functions

$$P_{\vec{n}_i,\vec{n}_j}(V_i,V_j) \tag{2.1}$$

for all possible vectors $\vec{n}_i$ and $\vec{n}_j$, where V_i and V_j are the values of the random variables $I(\vec{n}_i)$ and $I(\vec{n}_j)$, respectively. In most texture work and in all of the work in this thesis (except for the work in Chapter 7 and Chapter 8) the random field is assumed to be homogeneous, that is, all orders of probability densities are invariant through translations. Thus,

$$P_{\vec{n}_i,\vec{n}_j} \equiv P_{\vec{n}_i+\vec{c},\vec{n}_j+\vec{c}} \quad . \tag{2.2}$$

where $\vec{c}$ is an arbitrary vector constant. As an example,

$$P(V_1,V_2) \equiv P(V_3,V_4) \tag{2.3}$$

where V_1, V_2, V_3 and V_4 are as shown in Figure 2.1. In most of our work, dummy values of random variables (denoted for example by V_i) will be used to label pixels at vector location $\vec{n}_i$.

Given the assumptions that a texture field is homogeneous, the joint density functions $P_{\vec{r}}$ for all vector separations $\vec{r} = \vec{n}_i - \vec{n}_j$ represent the most complete set of second-order statistics possible. The statistical expectation of any functions of these joint density functions are called second-order statistics. If a pixel is connected to any of its neighbors on the same row, that is, if we consider neighbors immediately to the left or right (such as V_5 and V_6 in Figure 2.1), then their joint density is called a second-order nearest-neighbor joint density and any statistical expectations of the joint density are second-order nearest-neighbor statistics. Nearest-neighbor densities and nearest-neighbor statistics are very important in this chapter as the textures to be generated are primarily one-dimensional.

Similarly, third-order statistics are given by the set of third-order density functions

$$P_{\vec{n}_i,\vec{n}_j,\vec{n}_k}(V_i,V_j,V_k) \tag{2.4}$$

Assuming homogeneity of the texture, then

$$P_{\vec{n}_i,\vec{n}_j,\vec{n}_k} \equiv P_{\vec{n}_i+\vec{c},\vec{n}_j+\vec{c},\vec{n}_k+\vec{c}} \tag{2.5}$$

for all $\vec{n}_i$ and an arbitrary vector constant $\vec{c}$. As an example,

$$P(V_1,V_2,V_3) = P(V_4,V_5,V_6) \qquad (2.6)$$

in Figure 2.2. The statistical expectations of any function of these third-order densities are called third-order statistics. All second-order statistics may be derived from third-order joint densities. In this chapter, third-order statistics involving adjacent pixels along an image row, such as the pixels, V_7, V_8, V_9, in Figure 2.2, will be called third-order nearest-neighbor statistics.

Julesz [8] created computer generated patterns with controlled high-order statistical properties. A conclusion, often referred to as the Julesz conjecture, drawn from his work is that texture fields differing only in third- and higher-order statistics cannot be discriminated by a human viewer. Pollack [16] showed later that textures whose first- and second-order nearest-neighbor probabilities are equal may be discriminated by varying the third-order nearest-neighbor probabilities. Purks and Richards [3] extended this concept to create texture patterns that differ only in their statistics for four adjacent points. This study indicates that such textures can also be easily discriminated. However, as was pointed out by Pratt [14], the second-order probability densities of the two fields are not constrained to be equal for arbitrary pixel pairs along an image line. Thus there is still some question as to the relationships between measured mathematical parameters and human discriminability. Later work by Gagalowicz [5] seems to indicate that carefully generated binary patterns whose second-order probability pairs are equal for arbitrary distances can be visually discriminated by human observers and therefore presents a valid contradiction to the Julesz conjecture. Controlling different statistics of a texture is often a painfully difficult process and as a result, most of these textures are generated using approaches unlike the Markov approach of this chapter. Often blocks of pixels are generated with certain properties or patterns with special orientation and separation are used to study the effect of statistical changes on the human visual system. The mathematical relationships between the joint density functions of any texture are indeed complex.

For these reasons, we begin with one rather simple method of generating Markov one-dimensional binary textures. In later chapters, these ideas will be modified and extended to generate and simulate two-dimensional textures.

We have studied in detail the mathematical relationships of parameters involved in binary computer-generated one-dimensional texture patterns. Texture patterns in this paper have been generated using the mathematical relationships derived herein. Methods have been developed to control texture statistics for both nearest-neighbor and non-nearest-neighbor cases. Examples of both types of textures are presented. Using these methods, numerous counter examples to Julesz's conjecture may be generated and are illustrated in this Chapter.

Throughout this chapter, the $P(V_1,V_2,...,V_N)$ will denote the nearest-neighbor, Nth-order joint densities of our one-dimensional texture and the pixels $V_1,V_2,...,V_N$ will be adjacent to one another in the sequence as shown in Fig. 2.3 unless otherwise stated.

2.2 Generation Procedure

One-dimensional binary textures represent the simplest form of texture possible. It is believed that such binary patterns force human observers to utilize primitive visual mechanisms in discrimination. They are not designed to replace or imitate natural textures but are experimentally valuable in deriving concepts concerning texture attributes due to their mathematical simplicity.

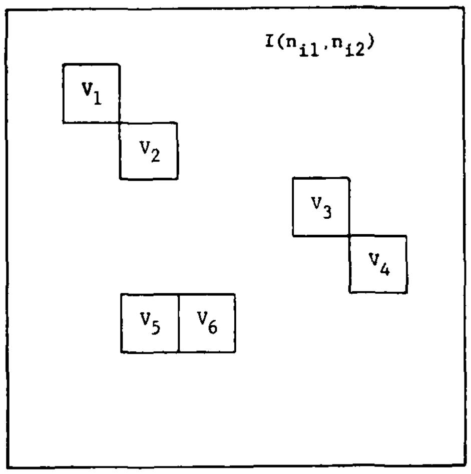


Figure 2.1 Second-order Statistics

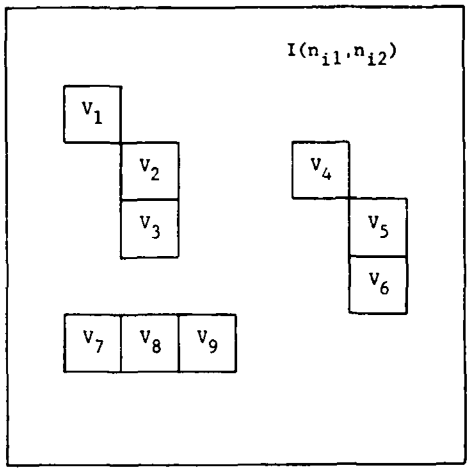


Figure 2.2 Third-order Statistics

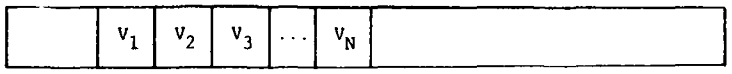


Figure 2.3 One-dimensional N-grams

In this experiment, binary one-dimensional sequences with carefully-controlled transition probabilities, dependent on the previous four points, were generated. Each sequence was then broken up into shorter 512-pixel strings which were stacked to form the two-dimensional 512x512 array which served as the texture pattern as is illustrated in Figure 2.4. Thus, the derived statistics are only controlled in one dimension but the final texture is two-dimensional.

We define the _a priori_ probability of a binary sequence of length N by $P(V_1,V_2,...,V_N)$ where each V_i, $i = 1,...,N$ is either 0 or 1. In our experiment this binary sequence is determined by generation parameters. This set of parameters, $G_0(V_1,V_2,...,V_N)$, each of which represents the probability of generating a 0 after the contiguous binary sequence $V_1,V_2,...,V_N$, defines the Markov process used to generate the texture pattern. It follows that the probability of generating a 1 after the sequence $V_1,V_2,...,V_N$ is $1-G_0(V_1,V_2,...,V_N)$. That is,

$$G_0(V_1,V_2,...,V_N) = 1-G_1(V_1,V_2,...,V_N) \quad (2.7)$$

Illustration of this commonly-used texture generation method is given by Purks and Richards [3]. However, it should be pointed out that their generation parameters were in many cases constrained to provide equal N-gram

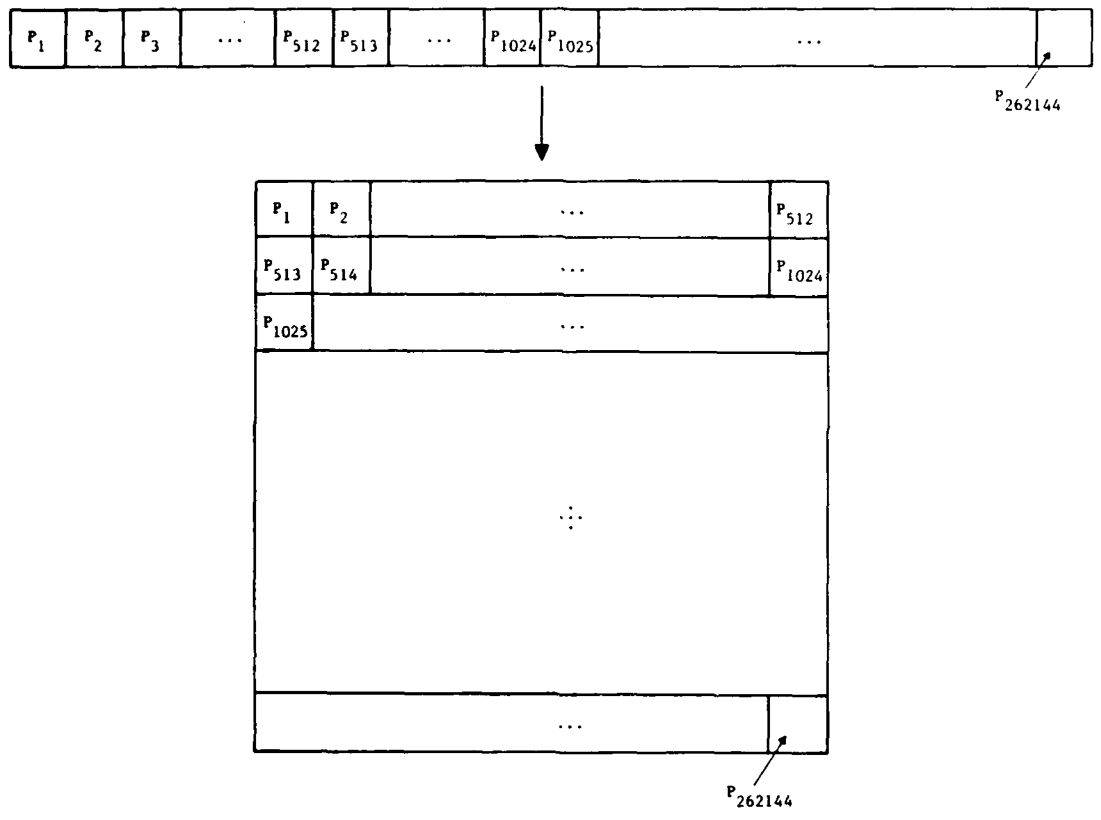


Figure 2.4 Forming a Two-Dimensional Image from a One-Dimensional Sequence

statistics, $P(V_1,V_2,...,V_N)$. The term "N-gram" refers to the Nth-order joint density function of a texture and was used by Purks and Richards in their study of texture. The term "N-gram"is frequently used in this thesis as a substitution for the longer terminology "Nth-order joint density function." A texture procedure, more general than that proposed by Purks and Richards, is detailed here.

These generation parameters are actually a special set of conditional probabilities. Only this specific group of conditional probabilities is used to generate the texture.

2.3 Analytical Analysis of the Markov Texture Process

By examining the mathematics of the Markov process we hope to generate patterns according to a set of given probabilities $P(V_1,V_2,...,V_N)$ which may be named the N-gram statistics of a specific pattern. We must therefore deal with the relationships that exist between these N-gram statistics and their generation parameters denoted by $G(V_1,V_2,...,V_N)$. Examining these relationships and also those between N-gram statistics of different lengths (that is the relationships between $P(V_1,V_2,...,V_{N_1})$ and $P(V_1,V_2,...,V_{N_2})$ for all N_1 and N_2) leads us to an understanding of the probabilistic system involved and thereby a method of generating desired texture patterns.

In generating random texture sequences, it is useful to first look at a simple analogy to the process from which basic concepts and conclusions can be drawn. We might regard this generation to be equivalent to the experiment consisting of tossing a "smart" coin that has a finite memory. In this case, $G_0(V_1,V_2,...,V_N)$ might represent the probability of tossing a "heads" given the previous sequence of N tosses was $V_1,V_2,...,V_N$. The resulting string of "1"'s and "0"'s (0 is the random variable denoting heads, 1 denotes tails) recorded from this experiment is our "texture." We realize immediately that the texture is "determined" by this set of generation parameters $G_0(V_1,V_2,...,V_N)$. Using the concept of conditional probability where P(A/B) is the probability of A given B we notice that

$$G_0(V_1,V_2,...,V_N) = P(0/V_1,V_2,...,V_N) \quad . \qquad (2.8)$$

Perhaps the most important concept derived from these generation parameters is that of the finite memory of the system. As is indicated by the notation $G_0(V_1,V_2,...,V_N)$, the probability of generating a zero depends on the string of binary values $V_1,V_2,...,V_N$ and not those "preceding" V_1. It is thereby suggested that our system has an N-gram memory and we will define such a system as

N-gram-dimensional. For example, returning to our coin tossing experiment, if we are in a four-gram-dimensional system, the probability of tossing a head depends on the four previous tosses only and all these conditional probabilities are determined by the sixteen parameters $G_0(V_1,V_2,V_3,V_4)$.

With these concepts in mind we can find these generation parameters $G_0(V_1,V_2,\ldots,V_N)$ given the desired probabilities $P(V_1,V_2,\ldots,V_N)$ and vice versa. The approach taken by Purks and Richards [3] in finding these N-gram statistics is based on sampling the generated textures. This may be seen by examining the entries in Table I of Ref. [3]. The entries correspond to the number of each N-gram counted in the texture generated and the accuracy of such probabilities depends on the law of large numbers. So the true probabilities $P(V_1,V_2,\ldots,V_N)$ are only approximated by the output textures and this approximation is poor when the physical size of the textures is small. These estimates have greater variance when the true probabilities are small. Therefore it is desirable to compute the exact probabilities given the generation parameters of the system.

Before proceeding further it is useful to prove the identity

$$P(V_1,V_2,\ldots,V_{N-1}) = P(V_1,V_2,\ldots,V_{N-1},0)+P(V_1,V_2,\ldots,V_{N-1},1) \tag{2.9}$$

Proof:

$$P(V_1,V_2,\ldots,V_{N-1},1) = P(V_1,V_2,\ldots,V_{N-1})*G_1(V_1,V_2,\ldots,V_{N-1}) \tag{2.10}$$

$$P(V_1,V_2,\ldots,V_{N-1},0) = P(V_1,V_2,\ldots,V_{N-1})*G_0(V_1,V_2,\ldots,V_{N-1}) = P(V_1,V_2,\ldots,V_{N-1})*(1-G_1(V_1,V_2,\ldots,V_{N-1}))$$

therefore

$$P(V_1,V_2,\ldots,V_{N-1},0)+P(V_1,V_2,\ldots,V_{N-1},1) = P(V_1,V_2,\ldots,V_{N-1}) .$$

As a result we have the following three sets of equalities

$$P(V_1,V_2,\ldots,V_{N-1},0) = P(V_1,V_2,\ldots,V_{N-1})*G_0(V_1,V_2,\ldots,V_{N-1}) \tag{2.11}$$

$$P(V_1,V_2,\ldots,V_{N-1},1) = P(V_1,V_2,\ldots,V_{N-1})*(1-G_0(V_1,V_2,\ldots,V_{N-1}))$$

$$P(V_1,V_2,\ldots,V_{N-1}) = P(V_1,V_2,\ldots,V_{N-1},0)+P(V_1,V_2,\ldots,V_{N-1},1) \tag{2.12}$$

$$G_0(V_1,V_2,\ldots,V_{N-1}) = P(V_1,V_2,\ldots,V_{N-1},0)/ (P(V_1,V_2,\ldots,V_{N-1},0)+P(V_1,V_2,\ldots,V_{N-1},1)). \tag{2.13}$$

Equation (2.13) results from Eqs. (2.11) and (2.12) and is essentially a statement of Bayes theorem for our problem.

2.4 Texture Statistics From Generation Parameters

Let us then consider the problem of obtaining the generation parameters (G's) from the N-grams (P's). Immediately we come to the conclusion that this is a trivial problem. One might merely use Eq. (2.13) to deduce the generation parameters. However, the equations derived so far do not indicate the extensive relationships which exist between the N-grams.

Equation (2.13) is not invertible so it is not useful in obtaining the N-gram statistics, $P(V_1,...,V_N)$ of a sequence given the generation parameters. As was stated above, once the generation parameters are defined, a texture may be generated using those parameters and the N-gram statistics are determined. We also know that once a complete set of N-gram statistics $P(V_1,...,V_{N_1})$ are defined for some N_1, the N-gram statistics $P(V_1,...,V_{N_2})$ may be resolved using Eq. (2.12) for all $N_2<N_1$. Given the generation parameters of a system, $G_0(V_1,V_2,...,V_N)$, we can analytically determine the N-gram statistics, $P(V_1,V_2,...,V_N)$ of the resulting texture.

The solution to this problem of finding N-gram statistics given generation parameters may be found by considering the generation procedure as a discrete state Markov process. This approach is readily seen when considering the generation parameters $G_0(V_1,V_2,\ldots,V_N)$ as transition probabilities. If we consider a two-dimensional system with P(0,0),P(0,1),P(1,0) and P(1,1) and generation parameters G(0,0), G(0,1), G(1,0) and G(1,1) we may define our system as composed of four possible states (0,0), (0,1), (1,0) and (1,1). If the system is in state i at the Kth observation and in state j at the (K+1)th observation then we say that the system has made a transition from state i to state j at the Kth stage of the generation process. In our example an observation is taken at each generation of a single new binary value and the state is determined by the values of the last two binary numbers generated. As an example, consider the sequence 0,1,1,0,0. We might consider the system to be in the (0,1) state at the start which may represent the Kth stage of our generation process then a transition is made to the (1,1) state at the (K+1) stage. These transition probabilities are determined by the generation parameters of the system. We also note that our N-dimensional system has 2^N possible states. As the transitions from each of these 2^N possible states to each of the 2^N possible states is fixed by our generation parameters we may form a transition matrix T whose elements t(i,j) represent the probability of a transition from the ith state to the jth state. If T is the transition matrix of a regular Markov chain, then there is a unique probability vector $\vec{p}$ which has positive coordinates and satisfies

$$T^T\vec{p} = \vec{p} \tag{2.14}$$

This same vector $\vec{p}$ may be computed by taking any row of the matrix

$$T^q \tag{2.15}$$

as q approaches infinity [17]. The vector $\vec{p}$ represents the vector of steady state probabilities. In our case it contains the desired probabilities $P(V_1, V_2, ..., V_N)$.

It is important to realize that this theorem holds for regular Markov processes. If there is an integer q such that every element of the matrix in Eq. (2.15) is stricly positive then the process is regular. Some processes are not regular such as absorbing Markov chains [17]. When any element of the transition matrix is equal to one along the diagonal, the process is said to be absorbing given that the system may begin in any state. This could happen if $G_0(0,0) = 1$ for example (a series of 0's would be generated in this case). For the purposes of our discussion we will assume that

$$0 < G_0(V_1, V_2, \ldots, V_N) < 1 \qquad (2.16)$$

for all V_i. This is a sufficient but not necessary condition for the process to be regular.

Applying these concepts to a two-dimensional system we obtain the transition matrix

$$\begin{array}{c} \\ \text{Initial} \\ \text{State} \end{array} \begin{array}{c} \text{Final State} \\ \left[\begin{array}{ccccc} & 00 & 01 & 10 & 11 \\ 00 & G_0(0,0) & 1-G_0(0,0) & 0 & 0 \\ 01 & 0 & 0 & G_0(0,1) & 1-G_0(0,1) \\ 10 & G_0(1,0) & 1-G_0(1,0) & 0 & 0 \\ 11 & 0 & 0 & G_0(1,1) & 1-G_0(1,1) \end{array}\right] \end{array} \quad (2.17)$$

The first row contains the transition probabilities from state (0,0) to states (0,0), (0,1), (1,0) and (1,1) in that order. The following set of equations results when Eq. (2.14) and Eq. (2.17) are combined:

$$\begin{bmatrix} G_0(0,0)-1 & 0 & G_0(1,0) & 0 \\ 1-G_0(0,0) & -1 & 1-G_0(1,0) & 0 \\ 0 & G_0(0,1) & -1 & G_0(1,1) \\ 0 & 1-G_0(0,1) & 0 & -G_0(1,1) \end{bmatrix} \times \begin{bmatrix} P(0,0) \\ P(0,1) \\ P(1,0) \\ P(1,1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.18)$$

As the above system is singular, we may form an equivalent non-singular set by replacing any equation with

$$P(0,0)+P(0,1)+P(1,0)+P(1,1) = 1 \qquad (2.19)$$

by using the fact that $\vec{p}$ is a probability vector. Solving this system gives the desired two-gram statistics $P(V_1,V_2)$.

Examining these generation parameters further we find that the same N-gram statistics may be generated by generation parameters of a different dimension. For example, consider the generation parameters in Table 2.1.

TABLE 2.1. EQUIVALENT GENERATION PARAMETERS

Set 1	Set 3	
$G_0(0,0) = 0.05$	$G_0(0,0,0) = 0.05$	$G_0(1,0,0) = 0.05$
$G_0(0,1) = 0.07$	$G_0(0,0,1) = 0.07$	$G_0(1,0,1) = 0.07$
$G_0(1,0) = 0.92$	$G_0(0,1,0) = 0.92$	$G_0(1,1,0) = 0.92$
$G_0(1,1) = 0.75$	$G_0(0,1,1) = 0.75$	$G_0(1,1,1) = 0.75$

Notice that the probability of generating a zero following a V_1,V_2,V_3 does not depend on V_1. The values, $G_0(V_1,V_2,V_3)$ of the second set indicate that the system is memoryless beyond two previous generation steps. We may write

$$G_0(V_1,V_2,V_3) = P(0/V_1,V_2,V_3) = P(0/V_2,V_3) = G_0(V_2,V_3) \tag{2.20}$$

for all V_2 and V_3.

It follows that, according to Eq. (2.11), the $P(V_1,V_2,\ldots,V_N)$ are also determined in our example for $N>2$ given $P(V_1,V_2)$ and $G_0(V_1,V_2)$. Thus we have

$$P(V_1,V_2,V_3) = P(V_1,V_2)*G_{V_3}(V_1,V_2) \tag{2.21}$$

$$\begin{aligned} P(V_1,V_2,V_3,V_4) &= P(V_1,V_2,V_3)*G_{V_4}(V_1,V_2,V_3) \\ &= P(V_1,V_2,V_3)*G_{V_4}(V_2,V_3) \end{aligned} \tag{2.22}$$

We conclude that given any set of generation parameters $G_0(V_1,V_2,\ldots,V_N)$ we may form a set of generation parameters $G_0(V_1,V_2,\ldots,V_N)$, M greater than or equal to N, according to the rule

$$G_0(V_1,V_2,\ldots,V_{M-N},V_{M-N+1},\ldots,V_M) = G_0(V_{M-N+1},\ldots,V_M) \tag{2.23}$$

that generate an equivalent set of N-gram statistics and therefore equivalent textures.

When desiring to generate textures according to a given set of N-gram statistics, $P(V_1,V_2,\ldots,V_N)$ we must realize the set of constraints imposed on the set. For example,

$$\sum_{V_1=0}^{1} \sum_{V_2=0}^{1} \cdots \sum_{V_N=0}^{1} P(V_1,V_2,\ldots,V_N) = 1 \qquad (2.24)$$

for all N. Returning to the set of equations used to determine the 2-gram statistics in matrix form we realize that, by adding the first two rows and last two rows of Eq. (2.18), $P(0,1) = P(1,0)$. In fact, by considering the set of equations arising from the set of equations derives from the generation systems of higher dimensions we find

$$\begin{aligned} &P(V_1,V_2,V_2,V_2,\ldots,V_2,V_3) = \\ &\quad P(V_3,V_2,V_2,V_2,\ldots,V_2,V_1) \quad . \end{aligned} \qquad (2.25)$$

This implies that many constraints are present on the N-gram statistics. For example, in the 3-gram-dimensional system containing $P(V_1,V_2,V_3)$

$$\begin{aligned} P(0,0,1) &= P(1,0,0) \\ P(0,1,1) &= P(1,1,0) \end{aligned} \qquad (2.26)$$

but also by Eq. (2.12) and the fact that $P(0,1) = P(1,0)$,

$$P(0,1,0)+P(0,1,1) = P(1,0,0)+P(1,0,1) \qquad (2.27)$$

and

$$P(0,0,1)+P(1,0,1) = P(0,1,0)+P(1,1,0)$$

By definition we also know that

$$P(0,V_1,V_2,\ldots,V_{N-1})*G_0(0,V_1,V_2,\ldots,V_{N-1})$$
$$+P(1,V_1,V_2,\ldots,V_{N-1})*G_0(1,V_1,V_2,\ldots,V_{N-1}) \quad (2.28)$$
$$= P(V_1,V_2,\ldots,V_{N-1},0)$$

and

$$P(0,V_1,V_2,\ldots,V_{N-1})*[1-G_0(0,V_1,V_2,\ldots,V_{N-1})]$$
$$+P(1,V_1,V_2,\ldots,V_{N-1})*[1-G_0(1,V_1,V_2,\ldots,V_{N-1})]$$
$$= P(V_1,V_2,\ldots,V_{N-1},1) \quad . \quad (2.29)$$

Combining Eqs. (2.28) and (2.29) with Eq. (2.12)

$$P(0,V_1,V_2,\ldots,V_{N-1})+P(1,V_1,V_2,\ldots,V_{N-1}) =$$
$$P(V_1,\ldots,V_{N-1}) \quad . \quad (2.30)$$

Equation (2.30) holds for all $N>2$.

Four constraining equations exist when Eqs. (2.24), (2.25) and (2.30) are reduced to orthogonal form for this 3-gram-dimensional system. This implies that we have four degrees of freedom when choosing the eight values $P(V_1,V_2,V_3)$ along with the obvious constraint that $P(V_1,V_2,V_3)>0$. This is precisely the number of degrees of freedom in the set of $G_0(V_1,V_2)$ which have only the range constraint of Eq. (2.16). For higher N-gram-dimensional systems, $P(V_1,\ldots,V_N)$ always has 2^{N-1} constraints on it from Eqs. (2.12), (2.24), (2.25) and (2.30). Thus we see that Eq. (2.13) holds in a degrees of freedom sense.

In conclusion, a method of determining N-gram statistics from generation parameters using the concept of a Markov chain was developed and a set of linear equations describing the constraints on the N-grams was presented. Using this approach to generating texture patterns a large variety of textures may be easily generated and examined using a minimum amount of effort.

2.5 Texture Moments

The above equalities and inequalities provide a full understanding of the texture generation system in probabilistic terms. Still further conclusions can be derived from them. From the above we see more clearly that the generation parameters $G(V_1,V_2,...,V_N)$ determine the texture completely and thus define the N-gram statistics $P(V_1,...,V_M)$ for all M. Also for a given set of $P(V_1,...,V_M)$ there can exist an infinite number of generation parameters, $G(V_1,V_2,...,V_N)$, which would generate many textures with such statistics if $N>M-1$. Provided the constraints on the statistics $P(V_1,...,V_M)$ are met just one texture could be generated if $N = M-1$ or perhaps none at all if $N<M-1$. Thus textures with equal first, second, third and fourth nearest-neighbor probabilities can be generated.

Parameters thought to be useful in texture discrimination may also be easily developed. For example, joint moments about the mean defined as

$$E[(x_1-\mu_1)^{r_1}(x_2-\mu_2)^{r_2}(x_3-\mu_3)^{r_3}\ldots(x_k-\mu_k)^{r_k}] \quad (2.31)$$

where $\sum_i r_i$ is the order of moment [18]. The rth moment of x_i is defined as

$$E(x_i^r) = \sum_{x_1}\sum_{x_2}\cdots\sum_{x_k} x_i^r f(x_1,\ldots,x_k) \quad (2.32)$$

where f is the joint probability distribution of the x_i. From our binary textures we could define the following parameters:

$$\begin{aligned}
\mu &= E\{x_0\} = \sum x_i f(x_i) = \sum x_i \; P(x_i) = \\
&\qquad 0\cdot P(0)+1\cdot P(1) = P(1) \\
\sigma^2 &= E\{(x_0-\mu)^2\} = \sum (x-\mu)^2 f(x) = \\
&\qquad (0-P(1))^2 P(0)+(1-P(1))^2 P(1) \\
&\qquad = P(1)-P(1)^2 \\
E\;\{(x_0-\mu)^3\} &= \sum (x-\mu)^3 f(x) = \qquad (2.33)\\
&\qquad (0-P(1))^3 P(0)+(1-P(1))^3 P(1) \\
&\qquad =2P(1)^3-3P(1)^2+P(1) \\
\alpha &= \frac{E\{(x_0-\mu)(x_1-\mu)\}}{\sigma^2} = \frac{P(1,1)-P(1)^2}{P(1)-P(1)^2}
\end{aligned}$$

$$\theta = \frac{E\{(x_0-\mu)(x_1-\mu)(x_2-\mu)\}}{\sigma^3} = \frac{P(1,1,1)-3P(1,1)\cdot P(1)+4P(1)^3}{(P(1)-P(1)^2)^{3/2}}$$

where $P(1)$, $P(1,1)$, $P(1,1,1)$ represent nearest-neighbor (N-gram) statistics although this can be changed to include non-nearest-neighbor statistics thus creating new texture parameters. The above parameters are useful in discrimination therefore only when textures differ in their (3-gram) or shorter statistics.

2.6 Constraining Second-Order Statistics

We describe now a method which allows non-nearest-neighbor statistics to be controlled using the relationships developed in the Sections 2.3 and 2.4. Because second-order probabilities are of interest we investigate the conditions required to assure equality of second-order statistics for non-nearest-neighbor statistics. If we denote $G_0(V_1,V_2,\ldots,V_M)$ as our generation parameters and $P(V_1,V_2,\ldots,V_N)$ to be their associated N-gram statistics then for the second-order statistics of one texture to be equal to another for the (N-1)st neighbor distance,

$$\sum_{V_2} \cdots \sum_{V_{N-1}} P_a(V_1, V_2, \ldots, V_M, V_{M+1}, \ldots, V_N) = \sum_{V_2} \cdots \sum_{V_{N-1}} P_b(V_1, V_2, \ldots, V_{M+1}, \ldots, V_N) \tag{2.34}$$

for $V_1, V_N \varepsilon \{0,1\}$ where P_a, P_b represents the N-gram statistics for the first and second texture respectively. It can also be shown that

$$\sum_{V_2} \cdots \sum_{V_{N-1}} P(V_1, V_2, \ldots, V_M, \ldots, V_N) = \sum_{V_2} \cdots \sum_{V_{N-1}} P(V_1, V_2, \ldots, V_M) \cdot \tag{2.35}$$

$$\prod_{k=M+1}^{N} [V_k + (-1)^{V_k} \cdot G_0(V_{k-M}, V_{k-M+1}, \ldots, V_{k-1})]$$

where

$$\sum_{V_j} = \sum_{V_j=0}^{1} \quad .$$

Recall that the N-gram statistics, P, are a function of the generating parameters G. If the second-order statistics are to be equal for two textures regardless of neighbor distance then Eq. (2.35) must hold for all N. It was previously believed that combining Eq. (2.34) and Eq. (2.35) yielded a non-redundant non-linear set of equations that would imply that two textures having equal second-order statistics must have the same generation parameters [19]. Julesz [8] also stated that Markov binary textures having equal second-order probability distributions cannot exist. He then proceeded to work with textures of four grey levels.

In the next section, we will show that binary textures with equal second-order statistics for all distances can be generated using a carefully-chosen set of generation parameters and that these texture contadict the Julesz conjecture.

2.7 Experimental Results

We may define second-order statistics for non-nearest-neighbors as

$$P(V_1,V_j) = \sum_{V_2} \cdots \sum_{V_{j-1}} \sum_{V_{j+1}} \cdots \sum_{V_N} P(V_1,V_2,\ldots,V_j,\ldots,V_N) \ . \qquad (2.36)$$

Purks and Richards [3] attempted to demonstrate that textures equal in second-order distribution could be generated with visually detectable differences. However, they merely held second- or third-order statistics equal between the two textures for small j, while varying second- and third-order statistics for longer j, that is, they allowed the second-order statistics for non-nearest-neighbors to be unequal over some distance. Examples of this type of manipulation are shown in

Figs. 2.5-2.10. The corresponding N-grams are shown in Table 2.2 and Table 2.3. The first column of each subtable contains the analytic N-grams for texture (a), top, and texture (b), bottom, which are found by solving Eq. (2.14) and Eq. (2.24) given the generation parameters in the third column. The middle column contains the actual N-gram statistics as measured from the generated textures themselves. The input, analytically-solved parameters will not be equal to the output statistics as the generation process is random. In other words, the output statistics are based on measurements taken on a sample from a population of textures having the characteristics defined by the input parameters.

Figures 2.5 and 2.6 show pairs of statistics having equal first- and second-order nearest-neighbor statistics. That is, $P_a(V_1,V_2) = P_b(V_1,V_2)$. There is also an internal equality for these textures which may be expressed as $P_a(V_1,V_2) = P_b(V_1,V_2) = 1/4$ for all V_1 and V_2 that are nearest-neighbors. Visual differences are apparent between the pairs. Figures 2.7 and 2.8 have texture pairs which have equal third-order, nearest-neighbor statistics both within and between the pairs, that is, $P_a(V_1,V_2,V_3) = P_b(V_1,V_2,V_3) = 0.125$. The bottom texture in Fig. 2.7 is a coin flip, purely random texture with the probability of both black and white equal. It is

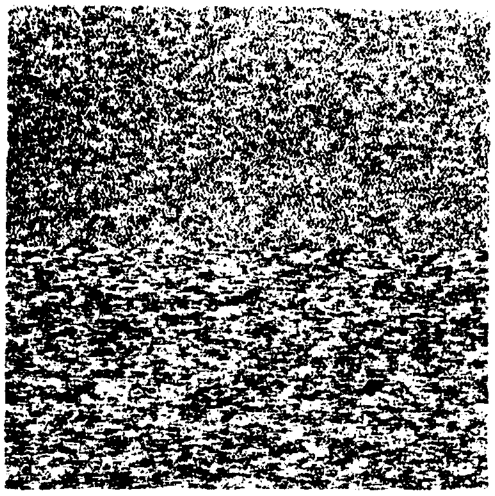


Figure 2.5 One-dimensional Texture

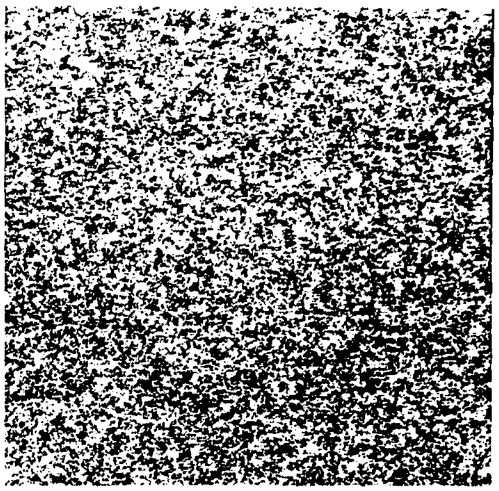


Figure 2.6 One-dimensional Texture

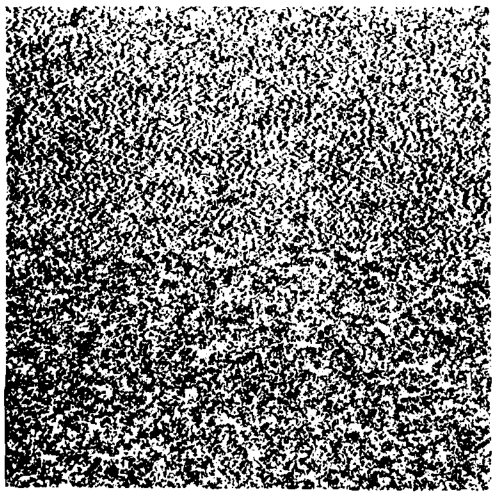


Figure 2.7 One-dimensional Texture

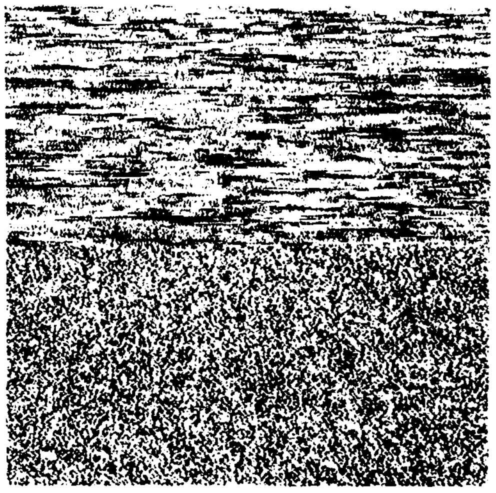


Figure 2.8 One-dimensional Texture

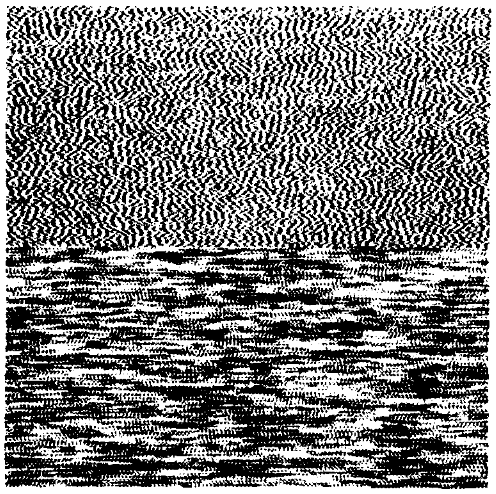


Figure 2.9 One-dimensional Texture

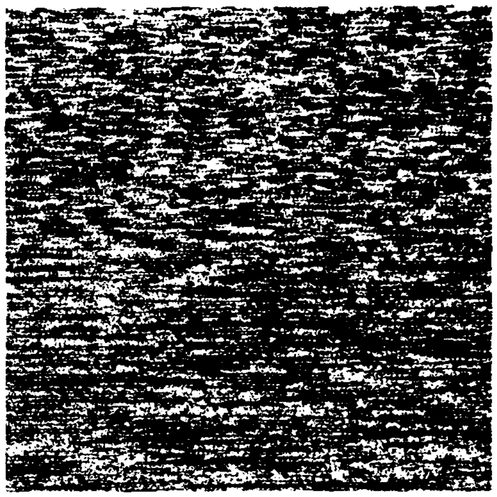


Figure 2.10 One-dimensional Texture

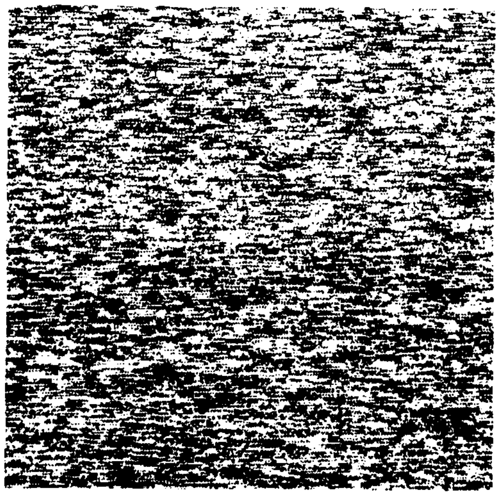


Figure 2.11 One-dimensional Texture

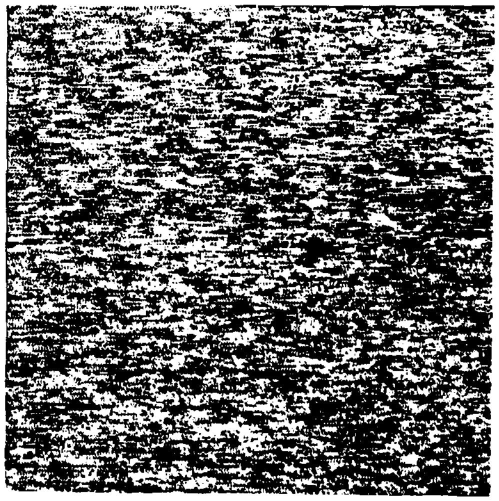


Figure 2.12 One-dimensional Texture

Table 2.2 N-GRAMS AND GENERATION PARAMETERS FOR FIGURES 2.5 - 2.8

STATISTICS FOR FIGURE 2.5:

	INPUT PARAMETERS		OUTPUT STATISTICS		GENERATION PARAMETERS	
	(a)	(b)	(a)	(b)	(a)	(b)
P(0)	.5000	.5000	.4983	.5009		
P(1)	.5000	.5000	.5017	.4991		
P(00)	.2500	.2500	.2478	.2504		
P(01)	.2500	.2500	.2505	.2505		
P(10)	.2500	.2500	.2505	.2505		
P(11)	.2500	.2500	.2512	.2486		
P(000)	.0033	.1667	.0814	.1663		
P(001)	.1667	.0033	.1664	.0042		
P(010)	.0033	.1667	.0035	.1679		
P(011)	.1667	.0033	.1670	.0026		
P(100)	.1667	.0033	.1664	.0042		
P(101)	.0033	.1667	.0041	.1663		
P(110)	.1667	.0033	.1670	.0026		
P(111)	.0033	.1667	.0042	.1650		
P(0000)	.0278	.1111	.0275	.1100	.5000	.7500
P(0001)	.0556	.0556	.0539	.0563	.5000	.7500
P(0010)	.0556	.0556	.0552	.0565	.7500	.5000
P(0011)	.1111	.0278	.1112	.0276	.7500	.5000
P(0100)	.0556	.0556	.0555	.0561	.5000	.7500
P(0101)	.0278	.1111	.0280	.1118	.5000	.7500
P(0110)	.1111	.0278	.1105	.0270	.7500	.5000
P(0111)	.0556	.0556	.0565	.0556	.7500	.5000
P(1000)	.0556	.0556	.0539	.0563	.2500	.5000
P(1001)	.1111	.0278	.1125	.0279	.2500	.5000
P(1010)	.0278	.1111	.0283	.1114	.5000	.2500
P(1011)	.0556	.0556	.0558	.0550	.5000	.2500
P(1100)	.1111	.0278	.1110	.0281	.2500	.5000
P(1101)	.0556	.0556	.0560	.0545	.2500	.5000
P(1110)	.0556	.0556	.0565	.0556	.5000	.2500
P(1111)	.0278	.1111	.0277	.1104	.5000	.2500

STATISTICS FOR FIGURE 2.6:

	INPUT PARAMETERS		OUTPUT STATISTICS		GENERATION PARAMETERS	
	(a)	(b)	(a)	(b)	(a)	(b)
P(0)	.5000	.5000	.4986	.4988		
P(1)	.5000	.5000	.5014	.5012		
P(00)	.2500	.2500	.2475	.2500		
P(01)	.2500	.2500	.2512	.2487		
P(10)	.2500	.2500	.2512	.2488		
P(11)	.2500	.2500	.2502	.2525		
P(000)	.1667	.0833	.1651	.0840		
P(001)	.0833	.1667	.0824	.1660		
P(010)	.0833	.1667	.0835	.1653		
P(011)	.1667	.0833	.1677	.0835		
P(100)	.0833	.1667	.0824	.1661		
P(101)	.1667	.0833	.1688	.0827		
P(110)	.1667	.0833	.1677	.0835		
P(111)	.0833	.1667	.0825	.1690		
P(0000)	.1111	.0278	.1108	.0283	.7500	.5000
P(0001)	.0556	.0556	.0543	.0556	.2500	.5000
P(0010)	.0278	.1111	.0278	.1104	.5000	.7500
P(0011)	.0556	.0556	.0546	.0557	.5000	.2500
P(0100)	.0278	.1111	.0279	.1101	.5000	.2500
P(0101)	.0556	.0556	.0555	.0552	.5000	.7500
P(0110)	.1111	.0278	.1128	.0272	.2500	.5000
P(0111)	.0556	.0556	.0549	.0563	.7500	.5000
P(1000)	.0556	.0556	.0543	.0556	.5000	.2500
P(1001)	.0278	.1111	.0281	.1104	.5000	.7500
P(1010)	.0556	.0556	.0557	.0549	.2500	.5000
P(1011)	.1111	.0278	.1131	.0278	.7500	.5000
P(1100)	.0556	.0556	.0545	.0559	.7500	.5000
P(1101)	.1111	.0278	.1133	.0275	.2500	.5000
P(1110)	.0556	.0556	.0549	.0563	.5000	.7500
P(1111)	.0278	.1111	.0276	.1127	.5000	.2500

STATISTICS FOR FIGURE 2.7:

	INPUT PARAMETERS		OUTPUT STATISTICS		GENERATION PARAMETERS	
	(a)	(b)	(a)	(b)	(a)	(b)
P(0)	.5000	.5000	.4995	.4996		
P(1)	.5000	.5000	.5005	.5004		
P(00)	.2500	.2500	.2500	.2488		
P(01)	.2500	.2500	.2496	.2508		
P(10)	.2500	.2500	.2496	.2508		
P(11)	.2500	.2500	.2509	.2496		
P(000)	.1250	.1250	.1251	.1235		
P(001)	.1250	.1250	.1249	.1253		
P(010)	.1250	.1250	.1244	.1260		
P(011)	.1250	.1250	.1251	.1248		
P(100)	.1250	.1250	.1249	.1253		
P(101)	.1250	.1250	.1247	.1255		
P(110)	.1250	.1250	.1251	.1248		
P(111)	.1250	.1250	.1257	.1249		
P(0000)	.0625	.0625	.0622	.0618	.5000	.5000
P(0001)	.0625	.0625	.0630	.0617	.0833	.5000
P(0010)	.0104	.0625	.0101	.0631	.5000	.5000
P(0011)	.1146	.0625	.1148	.0623	.5000	.5000
P(0100)	.1146	.0625	.1145	.0634	.5000	.5000
P(0101)	.0104	.0625	.0099	.0626	.5000	.5000
P(0110)	.0625	.0625	.0619	.0621	.0833	.5000
P(0111)	.0625	.0625	.0632	.0627	.5000	.5000
P(1000)	.0625	.0625	.0630	.0617	.5000	.5000
P(1001)	.0625	.0625	.0619	.0636	.0833	.5000
P(1010)	.1146	.0625	.1144	.0629	.9545	.5000
P(1011)	.0104	.0625	.0103	.0625	.5000	.5000
P(1100)	.0104	.0625	.0103	.0619	.5000	.5000
P(1101)	.1146	.0625	.1148	.0629	.9545	.5000
P(1110)	.0625	.0625	.0632	.0627	.0833	.5000
P(1111)	.0625	.0625	.0625	.0622	.5000	.5000

STATISTICS FOR FIGURE 2.8:

	INPUT PARAMETERS		OUTPUT STATISTICS		GENERATION PARAMETERS	
	(a)	(b)	(a)	(b)	(a)	(b)
P(0)	.5000	.5000	.4985	.4995		
P(1)	.5000	.5000	.5015	.5004		
P(00)	.2499	.2500	.2493	.2486		
P(01)	.2501	.2500	.2492	.2509		
P(10)	.2501	.2500	.2491	.2509		
P(11)	.2499	.2500	.2524	.2495		
P(000)	.1249	.1250	.1245	.1242		
P(001)	.1250	.1250	.1248	.1245		
P(010)	.1250	.1250	.1247	.1260		
P(011)	.1250	.1250	.1245	.1249		
P(100)	.1250	.1250	.1248	.1245		
P(101)	.1250	.1250	.1243	.1264		
P(110)	.1250	.1250	.1245	.1249		
P(111)	.1249	.1250	.1279	.1246		
P(0000)	.1145	.0104	.1140	.0101	.9545	.5000
P(0001)	.0104	.1146	.0105	.1140	.5000	.5000
P(0010)	.0625	.0625	.0524	.0630	.5000	.5000
P(0011)	.0625	.0625	.0624	.0615	.9167	.0833
P(0100)	.0625	.0625	.0520	.0625	.0833	.9167
P(0101)	.0625	.0625	.0527	.0635	.5000	.5000
P(0110)	.1145	.0104	.1140	.0105	.5000	.5000
P(0111)	.0104	.1146	.0104	.1144	.5000	.9545
P(1000)	.0104	.1145	.0105	.1140	.5000	.0455
P(1001)	.1146	.0104	.1143	.0104	.5000	.5000
P(1010)	.0625	.0625	.0523	.0630	.5000	.5000
P(1011)	.0625	.0625	.0621	.0634	.9167	.0833
P(1100)	.0625	.0625	.0528	.0620	.0833	.9167
P(1101)	.0625	.0625	.0617	.0629	.5000	.5000
P(1110)	.0104	.1146	.0104	.1144	.5000	.5000
P(1111)	.1145	.0104	.1175	.0103	.0455	.5000

Table 2.3 N-GRAMS AND GENERATION PARAMETERS FOR FIGURES 2.9 - 2.12

STATISTICS FOR FIGURE 2.9:

	INPUT PARAMETERS		OUTPUT STATISTICS		GENERATION PARAMETERS	
	(a)	(b)	(a)	(b)	(a)	(b)
P(0)	.5000	.5000	.4998	.5013		
P(1)	.5000	.5000	.5002	.4987		
P(00)	.2500	.2500	.2495	.2504		
P(01)	.2500	.2500	.2504	.2509		
P(10)	.2500	.2500	.2504	.2509		
P(11)	.2500	.2500	.2498	.2478		
P(000)	.1250	.1250	.1244	.1252		
P(001)	.1250	.1250	.1251	.1252		
P(010)	.1250	.1250	.1254	.1270		
P(011)	.1250	.1250	.1250	.1239		
P(100)	.1250	.1250	.1251	.1252		
P(101)	.1250	.1250	.1253	.1257		
P(110)	.1250	.1250	.1250	.1239		
P(111)	.1250	.1250	.1248	.1239		
P(0000)	.0625	.0625	.0623	.0620	.1000	.9000
P(0001)	.0625	.0625	.0620	.0632	.1000	.9000
P(0010)	.0625	.0625	.0627	.0636	.1000	.9000
P(0011)	.0625	.0625	.0625	.0616	.1000	.9000
P(0100)	.0625	.0625	.0629	.0633	.1000	.9000
P(0101)	.0625	.0625	.0624	.0637	.1000	.9000
P(0110)	.0625	.0625	.0626	.0614	.1000	.9000
P(0111)	.0625	.0625	.0624	.0625	.1000	.9000
P(1000)	.0625	.0625	.0620	.0632	.9000	.1000
P(1001)	.0625	.0625	.0631	.0620	.9000	.1000
P(1010)	.0625	.0625	.0627	.0634	.9000	.1000
P(1011)	.0625	.0625	.0625	.0623	.9000	.1000
P(1100)	.0625	.0625	.0622	.0619	.9000	.1000
P(1101)	.0625	.0625	.0628	.0620	.9000	.1000
P(1110)	.0625	.0625	.0624	.0625	.9000	.1000
P(1111)	.0625	.0625	.0624	.0614	.9000	.1000

STATISTICS FOR FIGURE 2.10:

	INPUT PARAMETERS		OUTPUT STATISTICS		GENERATION PARAMETERS	
	(a)	(b)	(a)	(b)	(a)	(b)
P(0)	.5625	.5626	.5623	.5650		
P(1)	.4375	.4374	.4377	.4350		
P(00)	.3800	.3802	.3800	.3828		
P(01)	.1825	.1824	.1822	.1822		
P(10)	.1825	.1824	.1822	.1822		
P(11)	.2550	.2549	.2555	.2528		
P(000)	.2975	.2977	.2975	.2994		
P(001)	.0825	.0825	.0825	.0833		
P(010)	.0825	.0825	.0821	.0829		
P(011)	.1000	.1000	.1002	.0993		
P(100)	.0825	.0825	.0825	.0833		
P(101)	.1000	.1000	.0998	.0989		
P(110)	.1000	.1000	.1002	.0993		
P(111)	.1550	.1549	.1553	.1535		
P(0000)	.2450	.2452	.2450	.2466	.8071	.9786
P(0001)	.0525	.0525	.0526	.0528	.7943	.3429
P(0010)	.0450	.0450	.0448	.0457	.8889	.9722
P(0011)	.0375	.0375	.0377	.0377	.4000	.0533
P(0100)	.0700	.0700	.0696	.0707	.6786	.7321
P(0101)	.0125	.0125	.0125	.0122	.9000	.7900
P(0110)	.0200	.0200	.0200	.0197	.5625	.1250
P(0111)	.0800	.0800	.0802	.0796	.1563	.1563
P(1000)	.0525	.0525	.0526	.0528	.9000	.1000
P(1001)	.0300	.0300	.0299	.0305	.1100	.9000
P(1010)	.0375	.0375	.0373	.0373	.8000	.7000
P(1011)	.0625	.0625	.0625	.0616	.0320	.2000
P(1100)	.0125	.0125	.0129	.0126	.4000	.1000
P(1101)	.0875	.0875	.0873	.0867	.3000	.3157
P(1110)	.0800	.0800	.0802	.0796	.0156	.1250
P(1111)	.0750	.0750	.0752	.0739	.9000	.9000

STATISTICS FOR FIGURE 2.11:

	INPUT PARAMETERS		OUTPUT STATISTICS		GENERATION PARAMETERS	
	(a)	(b)	(a)	(b)	(a)	(b)
P(0)	.5000	.5000	.4981	.4980		
P(1)	.5000	.5000	.5019	.5020		
P(00)	.2500	.2500	.2475	.2482		
P(01)	.2500	.2500	.2506	.2498		
P(10)	.2500	.2500	.2506	.2498		
P(11)	.2500	.2500	.2514	.2522		
P(000)	.2083	.0417	.2053	.0411		
P(001)	.0417	.2083	.0422	.2072		
P(010)	.0417	.2083	.0410	.2081		
P(011)	.2083	.0417	.2095	.0417		
P(100)	.0417	.2083	.0422	.2072		
P(101)	.2083	.0417	.2084	.0426		
P(110)	.2083	.0417	.2095	.0417		
P(111)	.0417	.2083	.0418	.2105		
P(0000)	.1736	.0069	.1704	.0065	.9000	.5000
P(0001)	.0347	.0347	.0350	.0345	.1000	.5000
P(0010)	.0069	.1736	.0071	.1726	.5000	.9000
P(0011)	.0347	.0347	.0350	.0346	.5000	.1000
P(0100)	.0069	.1736	.0069	.1724	.5000	.1000
P(0101)	.0347	.0347	.0341	.0356	.5000	.9000
P(0110)	.1736	.0069	.1743	.0070	.1000	.5000
P(0111)	.0347	.0347	.0352	.0347	.9000	.5000
P(1000)	.0347	.0347	.0350	.0345	.5000	.1000
P(1001)	.0069	.1736	.0072	.1727	.5000	.9000
P(1010)	.0347	.0347	.0339	.0355	.1000	.5000
P(1011)	.1736	.0069	.1745	.0071	.9000	.5000
P(1100)	.0347	.0347	.0353	.0348	.9000	.5000
P(1101)	.1736	.0069	.1743	.0069	.1000	.5000
P(1110)	.0347	.0347	.0352	.0347	.5000	.9000
P(1111)	.0069	.1736	.0066	.1759	.5000	.1000

STATISTICS FOR FIGURE 2.12:

	INPUT PARAMETERS		OUTPUT STATISTICS		GENERATION PARAMETERS	
	(a)	(b)	(a)	(b)	(a)	(b)
P(0)	.5000	.5000	.4998	.5009		
P(1)	.5000	.5000	.5002	.4991		
P(00)	.2500	.2500	.2496	.2527		
P(01)	.2500	.2500	.2503	.2482		
P(10)	.2500	.2500	.2503	.2482		
P(11)	.2500	.2500	.2499	.2509		
P(000)	.0616	.1884	.0602	.1906		
P(001)	.1884	.0616	.1895	.0620		
P(010)	.1884	.0616	.1890	.0605		
P(011)	.0616	.1884	.0613	.1877		
P(100)	.1884	.0616	.1894	.0620		
P(101)	.0616	.1884	.0608	.1862		
P(110)	.0616	.1884	.0613	.1877		
P(111)	.1884	.0616	.1885	.0632		
P(0000)	.0274	.1541	.0258	.1562	.7500	.9000
P(0001)	.0342	.0342	.0343	.0345	.4500	.2000
P(0010)	.1541	.0274	.1555	.0273	.9000	.7500
P(0011)	.0342	.0342	.0340	.0348	.2000	.4500
P(0100)	.1541	.0274	.1552	.0273	.1000	.2500
P(0101)	.0342	.0342	.0337	.0333	.8000	.5500
P(0110)	.0274	.1541	.0272	.1525	.2500	.1000
P(0111)	.0342	.0342	.0341	.0352	.5500	.8000
P(1000)	.0342	.0342	.0343	.0345	.2000	.4500
P(1001)	.1541	.0274	.1551	.0276	.9000	.7500
P(1010)	.0342	.0342	.0335	.0333	.4500	.2000
P(1011)	.0274	.1541	.0273	.1529	.7500	.9000
P(1100)	.0342	.0342	.0342	.0348	.5500	.8000
P(1101)	.0274	.1541	.0271	.1529	.2500	.1000
P(1110)	.0342	.0342	.0341	.0352	.8000	.5500
P(1111)	.1541	.0274	.1545	.0280	.1000	.2500

interesting to note that this texture is visually quite similar to the bottom texture of Fig. 2.6, yet the generation parameters and N-grams differ significantly. Thus two textures with differing N-gram statistics may be generated which are visually similar. Figure 2.9 shows a pair of textures which have equal fourth-order, nearest-neighbor statistics both within and between the pairs. That is,

$$P_a(V_1,V_2,V_3,V_4) = P_b(V_1,V_2,V_3,V_4) = 0.0625 \quad . \qquad (2.37)$$

The two are easily discriminated. Figure 2.10 shows two textures with between-equal, fourth-order nearest-neighbor statistics. The N-grams are not equal within however.

Figures 2.11 and 2.12 have texture pairs which are similar in many ways to the pair in Fig. 2.6. Both are counter examples to Julesz's conjecture that the eye is sensitive to only first-order and second-order probability distributions. Each has a texture pair where second-order statistics are equal for all distances between the texture pair. Precisely stated, $P_a(V_1,V_j) = P_b(V_1,V_j)$ for all j. These textures are generated using a set of restrictions discussed in Appendix A.

The textures in these figures are pseudo-randomly generated. Each texture was tested using a goodness-of-fit procedure to insure that textures with the desired N-grams were generated. This was done for 1-grams to 10-grams. The procedure is outlined in Appendix B. The goodness-of-fit depends on the pseudo-random number generator used, as a poor generator can be guaranteed to yield poor results in this type of experiment. The pseudo-random number generator used is detailed in Appendix C.

2.8 Conclusions

The one-dimensional binary patterns generated give rise to some basic concepts concerning textures and their discrimination. First of all they indicate the use of moments and similar statistics is not optimal at least in the nearest-neighbor sense as many textures have equal moments but are visually quite different. However, it should be pointed our that this may only be characteristic of some artificial textures and that moments could serve as good discrimination parameters in many real-world applications. Secondly, the results indicate a close relationship between second-order non-nearest-neighbor statistics and human discrimination. The counter-examples to the Julesz conjecture indicate that N-grams and higher-order statistics may be valuable in identifying and discriminating some textures. Use of these texture measures depends on factors such as discrimination accuracy desired, cost factors for statistics measure and the nature of the textures involved.

In later chapters, investigation of two-dimensional textures will be pursued as these correspond more closely to natural scene textures. The texture generation task will be approached from a simulation rather than a pure synthesis point of view. The complexity of controlling two-dimensional statistics in a synthesis process and attempts to study their effects on human discrimination and interpretation of textures is a problem which requires careful analysis and is beyond the scope of this work. It is very possible that no applicable, clear-cut results could be obtained from such a study. However, by simulating textures, models and processes for generating similar-looking textures are derived which may be useful in other applications and some simple statements concerning human discrimination of textures can also be made.

CHAPTER 3

TWO-DIMENSIONAL BINARY TEXTURE MODELS

3.1 Introduction

In this chapter, the concepts used to generate one-dimensional binary textures are extended to the two-dimensional case. This model is then used to simulate natural binary textures. A method for choosing the pixels in a non-contiguous generation kernel based on a linear model is described. This is an important concept in much of the work presented in this thesis. The method for collecting N-gram statistics is discussed and practical problems arising in this process are investigated. Results of natural texture simulation using this method are presented. Finally, the linear model which was used to determine the kernel pixels of greatest value in the N-gram generation process is used to generate binary simulations. This will lead us to the application of the linear model to continuous-tone textures in Chapter 5.

3.2 The 2-D Binary Markov Model

In the investigation of natural phenomenon once a researcher collects enough data he tries to imagine a process ..ich accounts for the results. The construction and development of a mathematical model is often the best way to do this. done. In some cases, the model may be extraordinarily complex, in others, exceedingly simple. In most cases model acceptance cannot be based on "truth" as the true generating phenomenon is too complex or simply unknown and so it is based upon model usefulness and "how well it works". Such "working" models for texture are presented in this chapter because they have the ability to simulate some natural textures.

If one can synthesize and simulate natural textures adequately by using some proposed model the criterion of usefulness and workability for that model is met. A researcher may then also apply the model to problems of texture identification and discrimination with justification. With any set of texture measures required for simulation, the information content of the measures is viaually indicated by the quality of the simulation and not merely by the percent of correct versus incorrect classifications when those textures are applied to the discrimination problem. By adding features important to texture simulation better methods of texture discrimination and identification can possibly be found.

Many early texture studies involved the use of binary textures generated by one-dimensional Markov processes. Such work was presented in Chapter 2. In these one-dimensional models a large vector of pixels was generated line by line using a set of generation parameters

$$G_{V_{N+1}}(V_1,V_2,\ldots,V_N)$$

where

$$G_{V_{N+1}}(V_1,V_2,\ldots,V_N) = P(V_{N+1}/V_1,V_2,\ldots,V_N) \quad (3.1)$$

and P(A/B) represents the probability of A given B. In the above notation each V_i represents a generated pixel which has value 0 (black) or 1 (white). Each pixel value, then, depends on the N pixels previous to it. A two-dimensional texture image is then formed by breaking up the large vector of pixels into shorter strings and stacking them one on top of the other (see Fig. 2.4). This procedure for large images nearly insured image row independence (unless N was large) thereby creating only horizontally oriented textures totally unsuitable for simulating natural two-dimensional textures.

By allowing N to increase exceeding the short string line length, two-dimensional (vertical and horizontal) dependence may be induced into the generating process. A pixel value then depends not only on the pixels previous to it on the same line but also on the pixels above it (see Fig. 3.1(b)). Thus, textures could be generated as a time sequence in television raster scan fashion. In theory, texture dependence could be extended ad infinitum, however practical considerations concerning the actual generation process show us that 2^N generation parameters must be accounted for. As a possible solution to the storage problem we can choose to ignore all but N of the previous pixels in our generation process and we can allow the pattern of the V_i's to become flexible. This idea will be discussed in later sections of this chapter.

Throughout the remainder of this thesis, the set of pixels, V_i's, on which the next pixel, V_{N+1}, depends will be referred to as the "kernel" of the synthesis process. The pixel V_{N+1} will be referred to as the "eye" of the kernel.

In order to estimate $P(V_1,V_2,..,V_N)$ for a fixed pattern $V_1,V_2,...,V_N$, all M substrings (samples) of length N are taken from a parent substring of length M+N-1 and the number of occurrences of the specific pattern $V_1,V_2,...,V_N$ are counted, then divided by M. This is equivalent to estimating the probability density function of a random variable by the histogram of a set of samples. Ignoring boundary conditions, linear unbiased estimates

($\hat{P}$'s) of the P's may be defined as

$$\hat{P}(V_1,V_2,\ldots,V_N) = \frac{1}{M}\sum_{j=1}^{M}\prod_{k=1}^{N}\delta(I(k+j)-V_k) \qquad (3.2)$$

where

$$\delta(V_j,V_k) = \begin{cases} 0 & \text{if } V_j \neq V_k \\ 1 & \text{if } V_j = V_k \end{cases}$$

and I(i) represents the ith element of the one-dimensional texture string from which the parameters are to be estimated. Equation (3.2) assumes that the V_i are contiguously located in order along a line.

This idea of estimating N-grams, $P(V_1,V_2,\ldots,V_N)$ from a sample parent texture may be extended to the two-dimensional case. A histogram of occurrences of each pattern of $(V_1,V_2,\ldots,V_N)$ is made by passing the two-dimensional kernel in Fig. 3.1(b) over the two-dimensional sample parent image. The tally is then divided by the total number of sample patterns observed to obtain $P(V_1,\ldots,V_N)$. As was stated earlier, two-dimensional synthesis is merely an extension of the one-dimensional case ignoring boundary conditions of the two-dimensional image.

Although it was not explicitly stated in Chapter 2, the generation parameters of a texture may be estimated for any given set of N V_i's from a parent texture. These statistics have the property

$$E[\hat{G}_{V_{N+1}}(V_1,V_2,\ldots,V_N)] = G_{V_{N+1}}(V_1,V_2,\ldots,V_N)$$

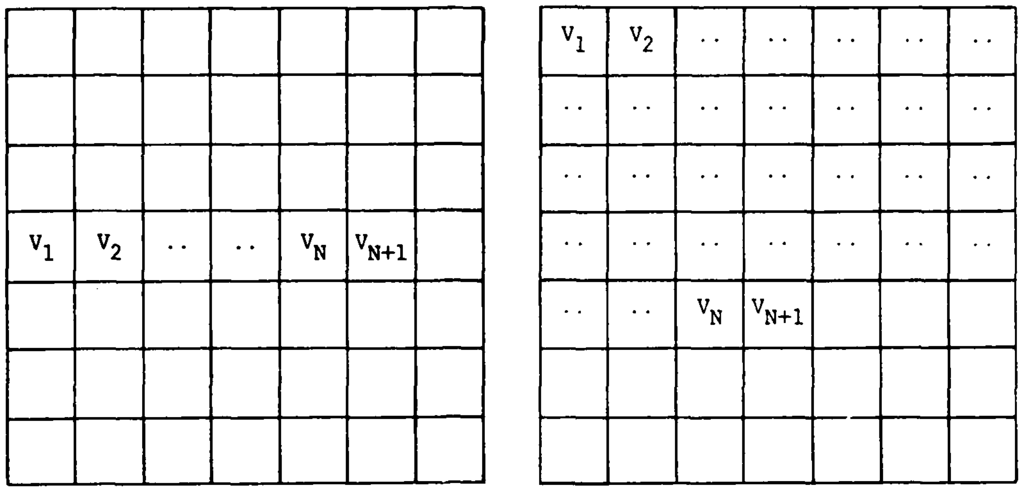
where

$$\hat{G}_{V_{N+1}}(V_1,V_2,\ldots,V_N) = \hat{P}(V_1,V_2,\ldots,V_N,V_{N+1})/ \\ (\hat{P}(V_1,V_2,\ldots,V_N,0)+\hat{P}(V_1,V_2,\ldots,V_{N,1})) \ . \tag{3.3}$$

3.3 Seeding the Generation Process

In the one-dimensional texture pattern generation of Chapter 2, a long binary vector of pixels is broken up into short strings which are stacked on top of each other (see Fig. 2.4). When synthesizing this one-dimensional string, a seed of four binary values is actually required to begin the process for a 4-gram-dimensional system. When the Markov process is regular and non-absorbing with none of the generation parameters equal to one or zero the process rapidly reaches a steady state independent of the seed.

A similar thing happens in the two-dimensional case but here the seed is larger and forms a two-dimensional frame around the synthesized image texture as shown in Fig. 3.2. With a two-dimensional texture generation kernel such as that shown in Fig. 3.1(b), the synthesized



(a) One-dimensional (b) Two-dimensional

Figure 3.1 Texture Synthesis Kernel

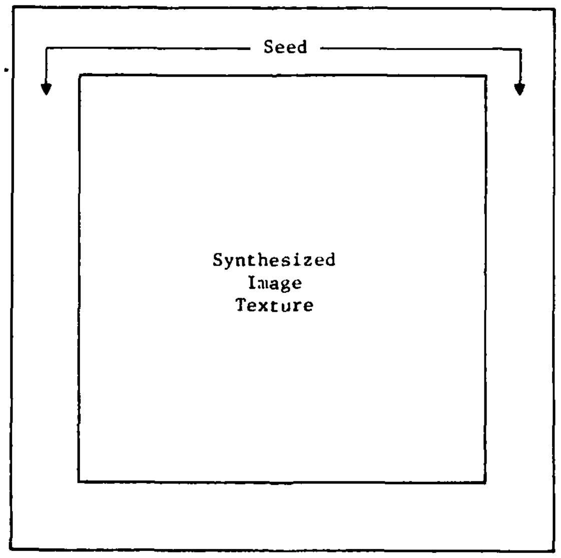


Figure 3.2 Synthesized Image Texture and Seed Region

image texture may be generated without using the bottom of the frame so the next pixel at each generation step depends only on pixels above it.

The seeding process may be handled in a variety of ways. The simplest approach would be to randomly generate the seed once for the whole image. In this case the pixel values in the seed frame of Fig. 3.2 remain the same throughout the generation process. A second approach would require the random generation of each pixel, V_i, in the generation kernel that fell outside the synthesized image texture region at each step during the generation process. Using this method, the pixel values in the seed frame change at each pixel generation. A third method would involve wrapping the image around such that the left edge of the synthesized image texture joins with the right side as in Fig. 3.3. In this case a random seed is required only to begin the process of the top of an image. A final method involves the use of another texture, usually a previously generated texture or the parent texture, as part of the seed, rather than noise. This method reduces the noise which otherwise occurs around the edges of the synthesized image texture.

Regardless of the seeding process, all texture synthesis methods developed in this thesis normally converge to a steady state within 5 to 20 pixels of the border of the image. This was confirmed by repeated studies of convergence effects on texture simulations. In most cases, this narrow region is not noticeable and is included as a part of the result. In some critical applications these edges could be thrown away.

3.4 Kernel Selection Using The Linear Model

We will refer to the V_i's on which the next pixel, V_{N+1}, depends as the kernel of the generation process. Geometrically speaking, the V_i's form a kernel "shape" or "pattern" which may or may not be spatially contiguous. For example, in Fig. 3.4 a generating kernel shape is shown where the V_5 pixel directly depends on only some of the pixels in its surrounding neighborhood. In this case, V_5 may be generated based on the values of pixels V_1, V_2, V_3 and V_4 but is directly dependent on no other pixels in the neighborhood. This does not imply that V_5 is not related or correlated with its other neighbors. In fact, the relationships between V_1, V_2, V_3, V_4, and V_5 will determine other interrelationships.

A non-contiguous neighborhood of V_i's is used as it allows a more parsimonious model for texture generation to be chosen. An analogy is in simple linear regression (as defined by Draper[20]) where independent variables which

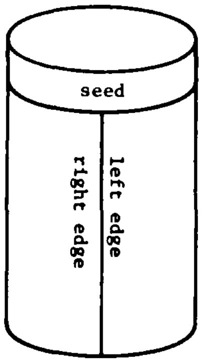


Figure 3.3 Wrap-around Texture Generation and Seed

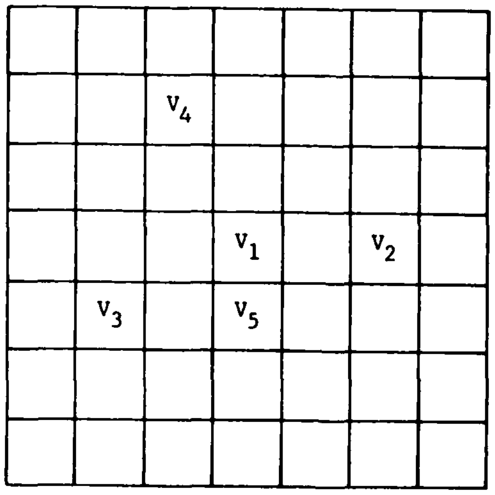


Figure 3.4 Two-dimensional Non-contiguous Texture Generation Kernel

do not contribute to the prediction or estimation of the dependent variable are dropped. In texture generation this allows the model to be estimated by fewer parameters and makes the generation-synthesis process more efficient by reducing the number of computations required. When generating textures based on N-grams, reducing N reduces the amount of storage required for 2^N generation parameters. By allowing the kernel of V_i's on which V_{N+1} depends to be non-contiguous, the range of dependence in a distance sense is increased over that which would be allowed with a contiguous kernel containing the same number of V_i's. This is very important to obtain the larger structure apparent in many textures. Reducing the number of pixels in the model also relieves us from the complex numerical problems of inverting matrices of unwieldy size, a necessary step in linear model parameter estimation discussed later in this chapter. We would, for example, not expect our V_{N+1} pixel to depend on a pixel V_i where the spatial separation between V_{N+1} and V_i is large. If that distance is small, however, we would expect a large dependence.

The method for choosing the proper independent variables (V_i's) to be included in the generation process requires special attention. We wish to choose the best subset of N variables from a larger finite neighborhood of

T variables, where $N \leq T$. Evaluating such subsets and their corresponding models requires a criterion. Texture results for each possible model could be visually examined and compared and the V_i's of the model corresponding to the visually most pleasing result could be chosen. However, $\binom{T}{N}$ model evaluations must be done using this approach. For a simple search through T = 40 points with N = 12, 5.5 billion models would have to be evaluated! This approach is therefore impractical and so a sub-optimal approach which yields a good but not necessarily the best set of V_i's for our model must be used.

If we view this problem as one of predicting a dependent variable, V_{N+1} from a large set of independent variables, V_i's, then the standard linear model approaches may be applied. In a statistical sense, independent variables are values that can be observed but not controlled and dependent variables are affected by changes in the independent variables. Thus the value of dependent variables is said to depend on values or changes in independent variables. The linear model is just one approach to explaining the relationship between independent and dependent variables.

In most linear model applications, the criterion used to evaluate the fit of the model to data is mean-square error. It is desirable in most cases to choose a model which minimizes the mean-square error. The problem in our case is to choose a subset of V_i's of size N from a set of V_i's of size T such that the linear model employing those V_i's produces the minimum mean-square error when compared to all other possible models containing N V_i's. This cannot be done without examining all $\binom{T}{N}$ models again - however, suboptimal approaches producing a very good, but not necessarily the best fit are available. One method employing a forward selection procedure is described in the following sections.

As a final note it should be observed that we are choosing the kernel for a non-linear N-gram-based on the fit of a linear model to sample data. This approach is admittedly <u>ad</u> <u>hoc</u> and is chosen for simplicity and computational ease. Regardless, it is believed that the kernel chosen by this approach is very good if not the best. If the value of a particular V_i is important to the prediction of V_{N+1} it will have a high partial correlation coefficient when it is examined for entry into the model during the forward selection proceedure. This is true in all cases when the pixels are binary-valued and is usually true when the pixels are continuously valued. Still, as we are not considering all $\binom{T}{N}$ possible sets of pixels in the neighborhood, the best model will not always be found [20].

The linear model which we use to determine the V_i's in the kernel may be expressed in linear regression form as

$$Y_k = \vec{X}_k^T \vec{\beta} + \varepsilon_k \qquad k = 1,2,\ldots,M$$

where

$$Y_k = V_{N+1,k} \qquad \vec{X}_k = \begin{bmatrix} 1 \\ V_{1,k} \\ V_{2,k} \\ \vdots \\ V_{N,k} \end{bmatrix} \tag{3.4}$$

$\vec{\beta}$ is an (N+1)x1 vector of unobservable parameters and ε_k is an unobservable random variable such that $E[\varepsilon_k] = 0$. The sample number is denoted by k and there are a total of M samples. We can also define matrices X and Y as

$$X = \begin{bmatrix} \vec{X}_1^T \\ \vec{X}_2^T \\ \vdots \\ \vec{X}_M^T \end{bmatrix} \qquad Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_M \end{bmatrix} \tag{3.5}$$

The most common estimator of $\vec{\beta}$ is $(X^TX)^{-1}X^TY$, the least-squares estimator. so given a parent texture wnich we desire to simulate, an estimation of the texture model parameter $\vec{\beta}$, $\hat{\vec{\beta}}$, may be obtained yielding the generation model

$$\hat{Y} = X\hat{\vec{\beta}} \tag{3.6}$$

It is very important to note that the estimation of the amount of noise present in a texture is usually obtained by measuring the amount of variation in the parent texture which is unexplained by our model. This is expressed numerically in the mean-square error. Therefore, as in any modelling process, any shortcomings or inaccuracies of the model will appear to be "noise" (unexplained variance) and hence the amount of noise will increase.

3.5 Correlation and Partial Correlation

We may define the mean and variance of a variable as

$$\mu_{V_1} = E[V_1] \tag{3.7}$$

and

$$\sigma^2_{V_1} = E[(V_1-\mu_{V_1})^2] \tag{3.8}$$

Similarly we may define the covariance of two variables V_1 and V_2 as

$$\begin{aligned}\gamma_{V_1V_2} &= E[(V_1-\mu_{V_1})(V_2-\mu_{V_2})] \\ &= E[(V_1V_2) - \mu_{V_1}\mu_{V_2}]\end{aligned} \tag{3.9}$$

And their correlation coefficient as

$$\rho_{V_1V_2} = \frac{\gamma_{V_1V_2}}{\sigma_{V_1}\sigma_{V_2}} \tag{3.10}$$

Using the above definitions we may then define the partial correlation coefficient of variables V_1 and V_2 after both have been adjusted for V_3 as

$$\rho_{V_1V_2 \cdot V_3} = \frac{\rho_{V_1V_2} - (\rho_{V_1V_3})(\rho_{V_2V_3})}{\sqrt{1-\rho_{V_1V_3}^{2}}\sqrt{1-\rho_{V_2V_3}^{2}}} \tag{3.11}$$

Each of the above parameters has a corresponding statistic (estimate of a parameter of a population given an observable sample of that population) which is chosen to meet some desirable set of a criteria such as sufficiency, consistency and unbias of estimate under certain conditions. Given that the $V_{i,k}$'s are samples from an ith order multivariate normal distribution the maximum likelihood estimates for the above parameters are

$$\hat{\mu}_{V_1} = \frac{\sum_k V_{1,k}}{M} = \bar{V}_1 \tag{3.12}$$

$$\hat{\sigma}_{V_1}^2 = \frac{\sum_k (V_{1,k} - \bar{V}_1)^2}{M} \tag{3.13}$$

$$r_{V_1V_2} = \frac{\sum_k (V_{1,k}-\bar{V}_1)(V_{2,k}-\bar{V}_2)}{\left\{\left[\sum_k (V_{1,k}-\bar{V}_1)^2\right]\left[\sum_k (V_{2,k}-\bar{V}_2)\right]^2\right\}^{1/2}} \qquad (3.14)$$

$$r_{V_1V_2\cdot V_3} = \frac{r_{V_1V_2}-(r_{V_1V_3})(r_{V_2V_3})}{\sqrt{1-r^2_{V_1V_3}}\sqrt{1-r^2_{V_2V_3}}} \qquad (3.15)$$

Second-order partial autocorrelation may be found in a similar manner using

$$r_{V_1V_2\cdot V_3V_4} = \frac{r_{V_1V_2\cdot V_3}-r_{V_1V_4\cdot V_3}r_{V_2V_4\cdot V_3}}{\sqrt{1-r^2_{V_1V_4\cdot V_3}}\sqrt{1-r^2_{V_2V_4\cdot V_3}}} \qquad (3.16)$$

Higher-order partial correlations may be found by extension of the above [21].

3.6 The Forward Selection Procedure

One approach to finding a good subset of independent variables is known as the forward selection procedure. This method inserts N variables into the model one-at-a-time. The order of insertion is determined by using the partial-correlation coefficient as a measure of the importance of variables not yet in the equation. The basic p edure is as follows. First we select the V_i most correlated with V_{N+1} denoted as V_{i_1}, and we find the first-order linear regression equation $\hat{V}_{N+1} = a_1 V_{i_1} + a_2$. We next find the partial correlation coefficient of V_j (for all $j \neq i$) and V_{N+1} (after allowance for V_{i_1}). Mathematically this is equivalent to finding the correlation between the residuals from the regression $\hat{V}_{N+1} = a_1 V_{i_1} + a_2$ and the residuals from another regression $V_j = b_1 V_{i_1} + b_2$ (which we have not actually performed). The V_j with the highest partial correlation coefficient with V_{N+1}, V_{i_2}, is now selected and a second equation $\hat{V}_{N+1} = c_1 V_{i_1} + c_2 V_{i_2} + c_3$ is fitted to the data. This process continues. After $V_{i_1}, V_{i_2}, \ldots, V_{i_q}$ are in the regression the partial correlation coefficients are the correlations between the residuals from the regression $\hat{V}_{N+1} = f(V_{i_1}, V_{i_2}, \ldots, V_{i_q})$ and the residuals from a regression $V_{i_j} = f_j(V_{i_1}, V_{i_2}, \ldots, V_{i_q})$ $(j \geq q)$. The V_{i_j} with the highest partial correlation coefficient is now selected for entry into the linear model. The process is continued until N V_i's are entered into the model.

The final N variables chosen by a forward selection procedure are not guaranteed to be the optimal set but given the logistics of the selection procedure, the solution obtained is usually close to optimal.

One of the most common procedures for implementing the forward selection process numerically utilizes Doolittle decomposition [22]. The Doolittle decomposition may be used to find the inverse of the correlation matrix, R_p, and the estimates of linear model coefficients as each variable is entered in the model. The correlation matrix merely consists of the set of correlation coefficients $r_{V_iV_j}$ as defined by Eq. (3.14). It is then factored into the product of two triangular matrices

$$R_p = L_pU_p \tag{3.17}$$

where L_p is lower triangular and U_p is upper triangular with ones on the diagonal. Partial autocorrelation coefficients may be obtained easily from elements of this matrix during its decomposition at each step. For further details and examples on the forward selection procedure of the decomposition process see Beyer [18] and Draper and Smith [20].

3.7 Statistics Collection

In practice, N, the number of pixels in the generation kernel, is often chosen by computational limits imposed by finite processing capability or finite computer memory. The idea of parsimony would also urge the selection of the smallest N possible. In the most general texture synthesis (stochastic) model which utilizes N-grams we find that as many as g^N storage locations are required in order to collect the data needed to synthesize a texture from a parent where g is the number of grey levels in the texture. This approach calls for N to be less than 17 for $g = 2$ (a binary image) if we have only $2^{17} = 131,072$ storage locations in memory. Approaches to "stretch" this limitation have been investigated and will be discussed in a latter section. If we have 8 grey levels then N must be less than or equal to 5 as $8^5 = 32,768$. Thus, in synthesizing textures using an N-gram approach, processor storage capability is the major limiting factor.

Determining which pixels will be included in the generation kernel requires an estimate of the linear model defined in Eq. (3.4). To do this the X and Y matrices of Eq. (3.5) may be used or the correlation matrix of the kernel points must be estimated using a parent texture.

The elements of the $\vec{X}_k$ vector of Eq. (3.4) are obtained by passing the kernel window over the sample parent texture and recording the pixel value corresponding to the position of each V_i in the window. The kernel is assumed to be completely within the boundary of the parent texture. A sample may be taken at all possible positions in the parent texture or a random sample of points may be chosen if the parent texture is very large to reduce the number of computations required. In actual practice, no X matrix is ever formed. To obtain the correlation matrix of the sample set, R, (and from that $\hat{\beta}$) we need only the sum of squares cross products, V_i and V_j, over the sample set according to Eq. (3.14).

The elements of the correlation matrix for many kernel patterns often contain redundancies. For example the spatial relationship between the pair V_1 and V_2 and the pair V_3 and V_5 in Fig. 3.4 is the same. Estimating the correlation for these two pairs of points from a sample will yield nearly equal values if the sample size is large and the overlap of samples used to estimate the correlation values is large. In short,

$$r_{V_1V_2} \cong r_{V_3V_4} \tag{3.18}$$

In more mathematical terms, let $I(n_1,n_2)$ denote the random texture field where n_1 and n_2 are integers representing the coordinates of points in our sampled and quantized image. Let $\vec{n}$ be the vector of coordinates (n_1,n_2). From Section 2.1, second-order or 2-gram statistics are given by the set of joint density functions

$$P_{\vec{n},\vec{m}}(V_i,V_j) \tag{3.19}$$

for all possible vectors $\vec{n}$ and $\vec{m}$, where V_1 and V_2 are the pixel values of the random variables $I(\vec{n})$ and $I(\vec{m})$, respectively. If the random field is homogeneous, that is, invariant through translations then

$$P_{\vec{n},\vec{m}} = P_{\vec{n}-\vec{m},\,\vec{0}} \tag{3.20}$$

from Eq. (2.2).

If we assume that our two-dimensional texture is homogeneous and stationary we might propose that Eq. (3.20) is true by definition. A word of caution is necessary at this point. Estimating the elements of the covariance or correlation matrix using the assumption of stationarity and homogeneity can yield correlation matrices having negative determinants, in violation of the fact that non-singular covariance matrices must be symmetric positive definite. The violation occurs because the sample is not homogeneous and stationary. Therefore the type of assumption expressed in Eq. (3.20) should only be used for estimation when sample sizes are very large and are known to be homogeneous and stationary.

The assumption of Eq. (3.20) can be very powerful in a computational sense for large samples as the calculation of covariance (or correlation) matrices can be time consuming. The complete covariance matrix for a contiguous 10 x 10 (100 element) kernel contains 10,000 entries. Of these, 4050 are redundant by simple symmetry as $\sum V_i V_j = \sum V_j V_i$ which implies $r_{V_i V_j} = r_{V_j V_i}$. Computing all cross products and sums to estimate this matrix from a 512 x 512 image requires over 1.025 billion multiplications and 1.05 billion additions. Utilizing the concept of homogeneity, this may be reduced to 0.05 billion multiplications and 0.076 billion additions which is significantly lower.

Once the covariance matrix of a kernel is determined, the linear model containing the V_i's of the kernel may be obtained. It is advisable to make the kernel large as more texture information is contained over large distances. In many cases, there are relationships between pixels separated by great distances especially if the texture is coarse or highly regular (periodic). In practice, however, large matrices may be numerically ill-conditioned during a decomposition or inversion process and so they must be avoided. Also computer storage limitations must be considered. In this study, no matrix larger than 100x100 was decomposed or inverted for these reasons. This constraint required a multiple-pass approach.

First, 100 V_i's (usually closest to the kernel eye,

V_{N+1}) were chosen for examination. They were entered into the linear model in a forward selection manner until it was determined that entry of additional V_i's would be insignificant. Insignificance is indicated both statistically by considering the reduction in the sum of squares due to the entry of a variable into the model [20] and also by the value of $\hat{\beta}_i$ which approaches zero so the variable V_i becomes insignificant. On the second pass, variables not tried in the first pass are examined and tested for possible entry into the model. Again, those which are significant are kept and the others are discarded. On the next pass, any variables not examined in the first two passes may be examined. The process continues until all V_i's have been tested for possible entry into the linear model at least once.

This multiple-pass process, as the forward selection procedure which it employs, is not guaranteed to produce the best model but will provide an excellent model nevertheless. The final model will also be parsimonious. This will reduce the number of parameters in our model and aid in the efficiency of the generation process.

3.8 Results

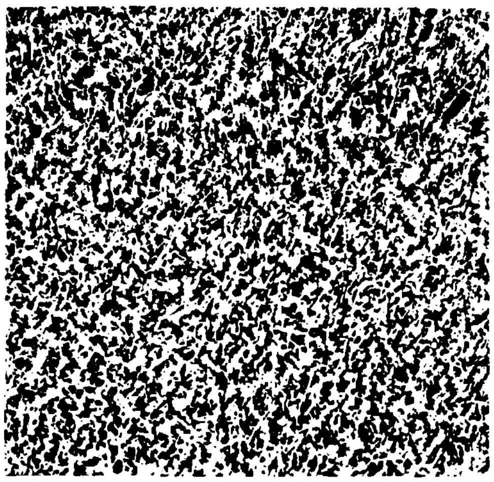
Once the points for the kernel are chosen based on the linear model derived using the methods described in the previous sections, estimates of the generation parameters for the texture are obtained using concepts discussed in section 3.2. Practical considerations require us to limit N, the number of pixels in the kernel, to 12 to 18 depending on the processor storage available as 2^N values must be stored. These $\hat{G}$'s are then used to generate each pixel along a row, row by row until a complete two-dimensional texture is obtained. For each pixel the appropriate generation parameter estimate is found and a uniformly-distributed pseudo-random variable is generated. Based on these two values, a black pixel (0) or white pixel (1) is generated.

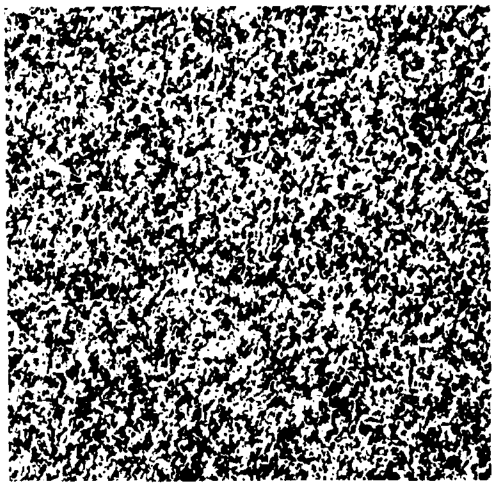
In practice, not all of the generation parameters may be estimated when N is large because all possible patterns of $V_1, V_2, \ldots, V_N, V_{N+1}$ may not be present in the sample image or there may be few of them. Smaller samples can cause inaccurate estimation of the G's as the variance of our estimate is larger and therefore the expected error of our estimate is larger than would be expected with a larger sample size. In these cases it is important to sum over the least significant kernel elements and estimate $G_{V_{N+1}}(V_1, V_2, \ldots, V_N)$ by $\hat{G}_{V_{N+1}}(V_i, V_{i+1}, \ldots, V_N)$. In our study, this was done if the sample size to compute $\hat{G}_{V_{N+1}}(V_1, V_2, \ldots, V_N)$ was less than 10. The variable i is increased until this condition is met.

Texture simulations using this method are shown in Figs. 3.5(b)-3.14(b). Visually, the results are very good. As the estimated texture generation parameters are approximated using statistics gathered from the full parent texture, non-homogeneity in the parent texture will cause an "average" texture to be synthesized. The simulation of straw (Fig. 3.7) is poor because of its non-homogeneous nature (specifically the directionality of the stalks in different parts of the image) and exhibition of detail (specifically individual non-conforming single stalks). A similar observation may be made with respect to the parent textures of grass and water but in these textures the non-homogeneity is not so pronounced. As we are attempting to synthesize textures and not merely "image code" the parent textures, details and non-homogeneities will be lost in the synthesis process.

The bark texture is among the most difficult to simulate due to its very unusual macro-structure. Still, the N-gram simulation looks remarkably similar to the original when windowed regions 20 to 40 pixels square are observed.

The kernels used to generate these N-gram simulations are shown in Table 3.1. A clear-cut relationship between the textures and the kernels found using the linear model


(a) Original Texture


(b) N-gram Simulation

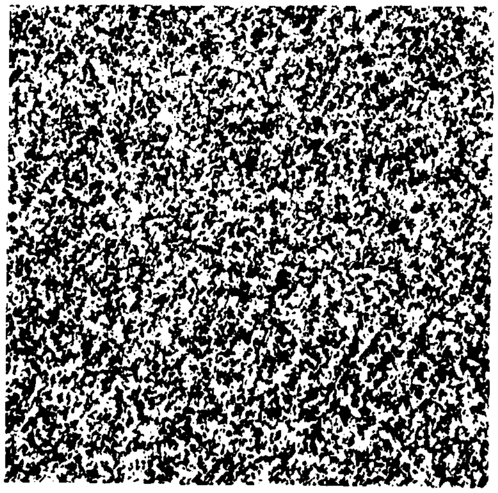
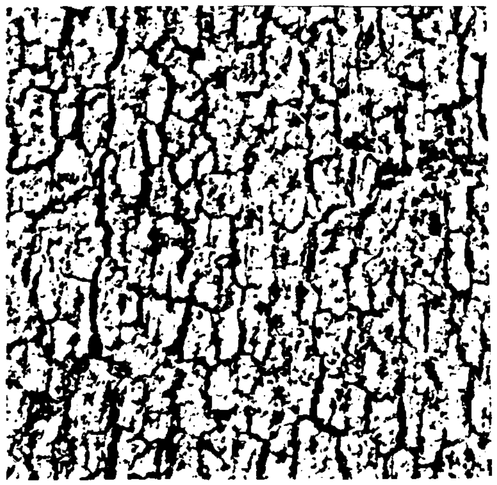
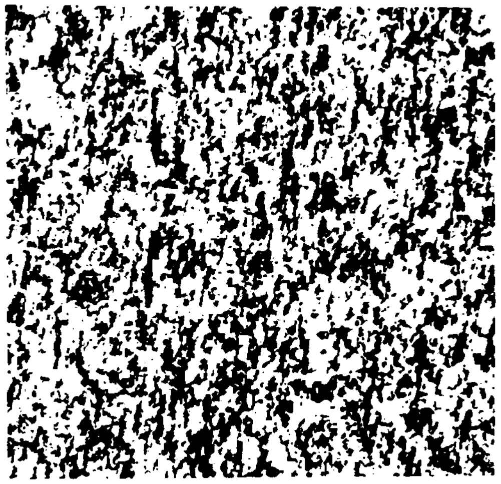

(c) Linear Model Simulation

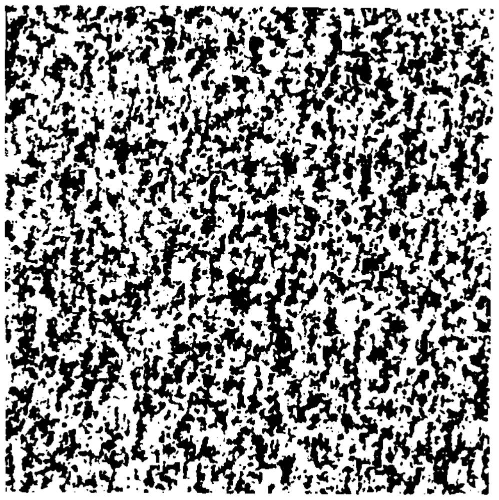
Figure 3.5 Grass



(a) Original Texture

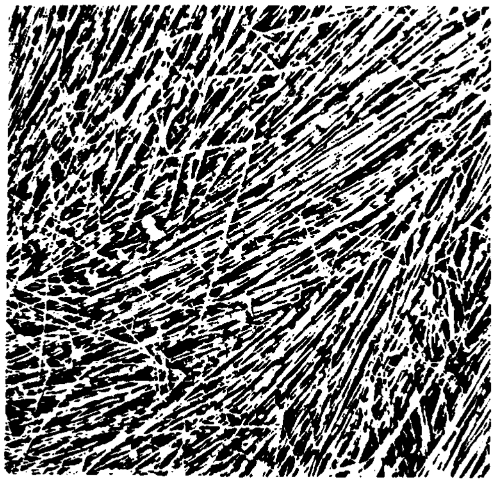


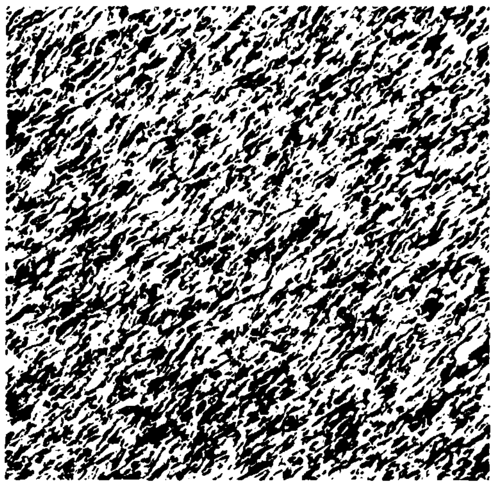
(b) N-gram Simulation



(c) Linear Model Simulation

Figure 3.6 Bark


(a) Original Texture


(b) N-gram Simulation

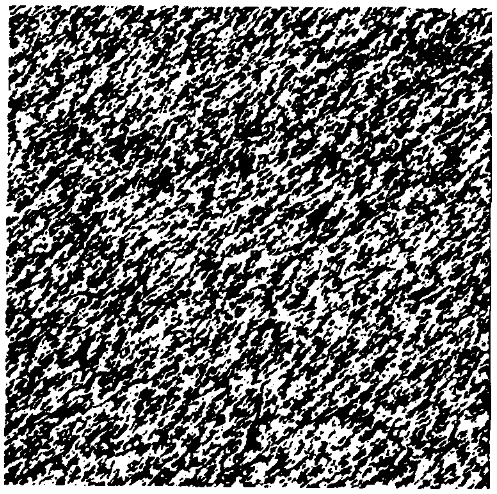
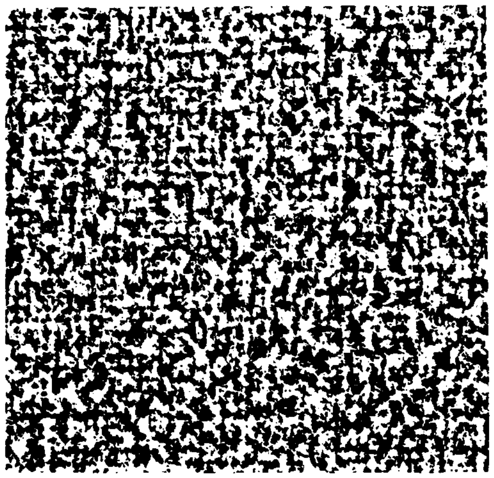
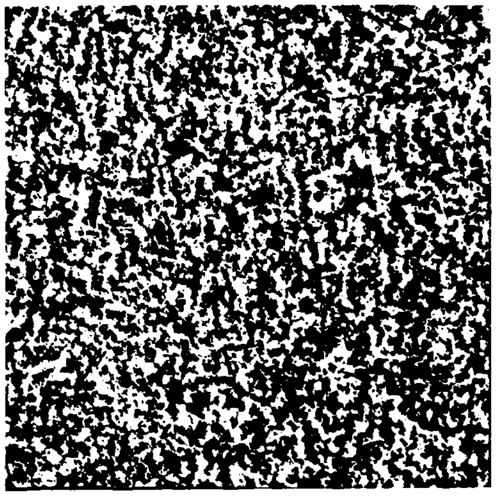

(c) Linear Model Simulation

Figure 3.7 Straw


(a) Original Texture


(b) N-gram Simulation

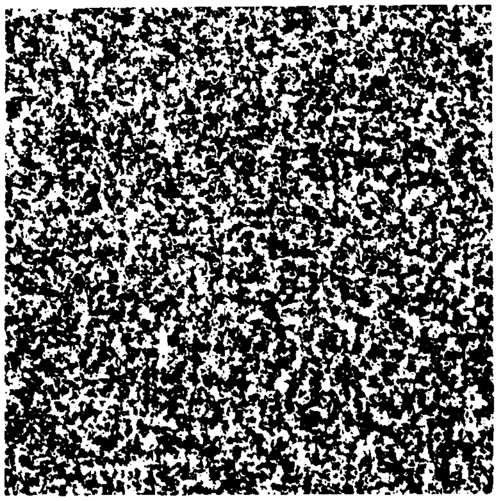
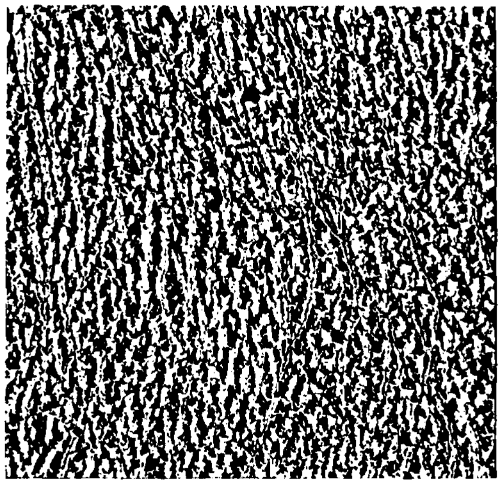
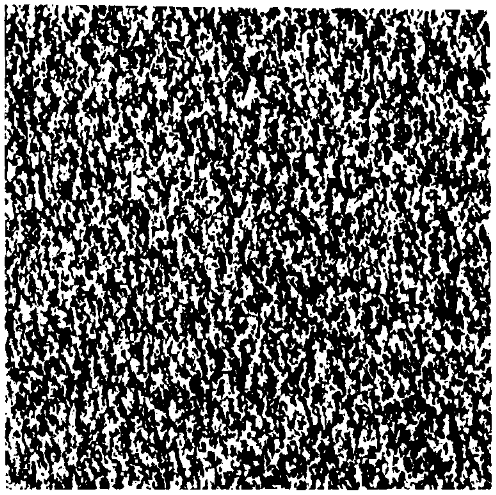

(c) Linear Model Simulation

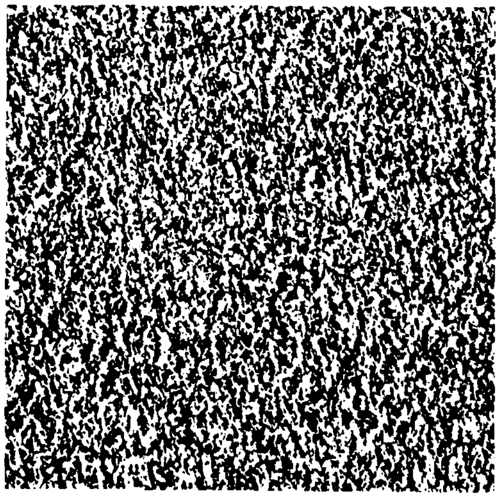
Figure 3.8 Wool



(a) Original Texture

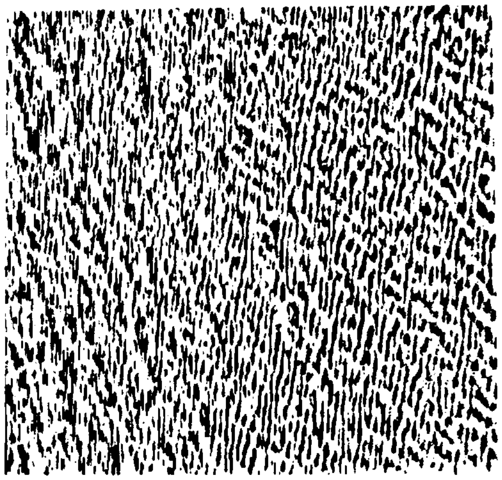


(b) N-gram Simulation

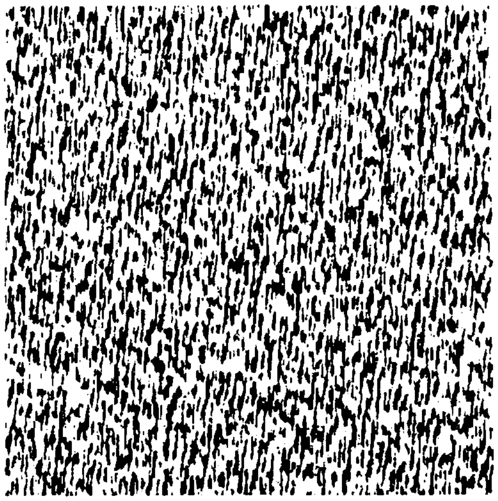


(c) Linear Model Simulation

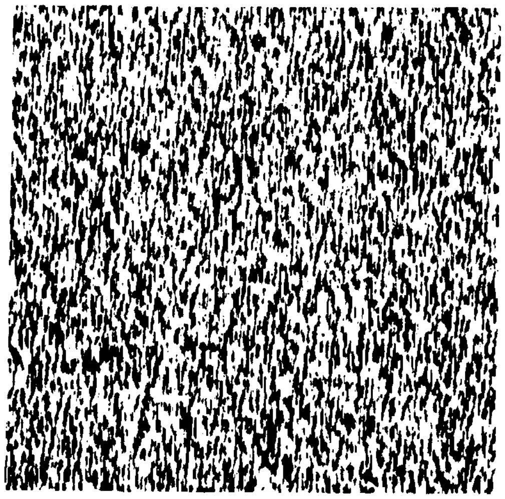
Figure 3.9 Leather



(a) Original Texture

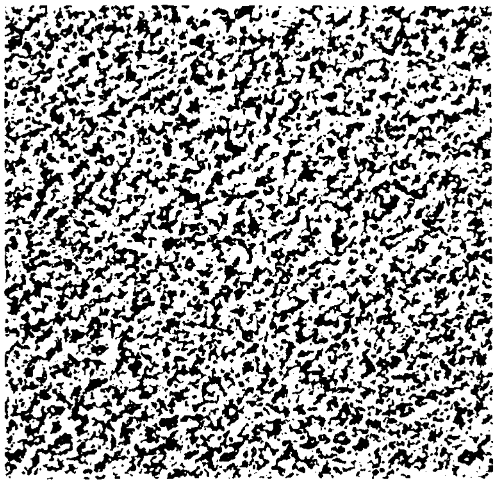


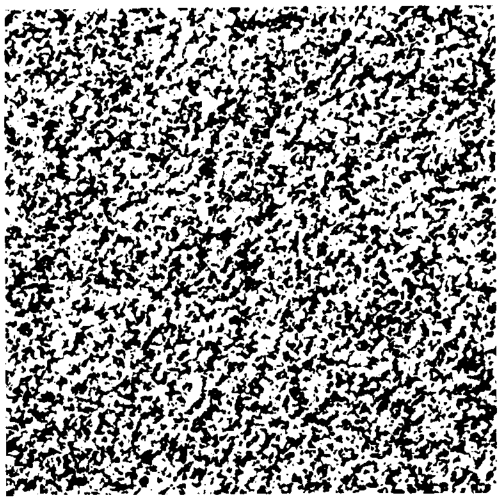
(b) N-gram Simulation



(c) Linear Model Simulation

Figure 3.10 Water


(a) Original Texture


(b) N-gram Simulation

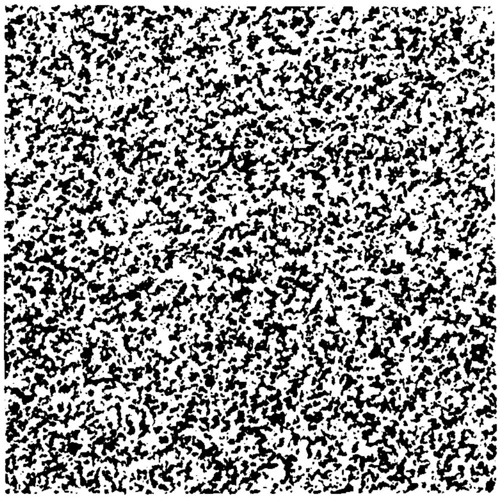
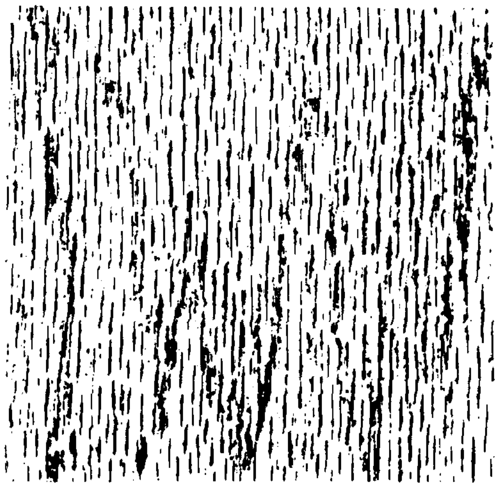
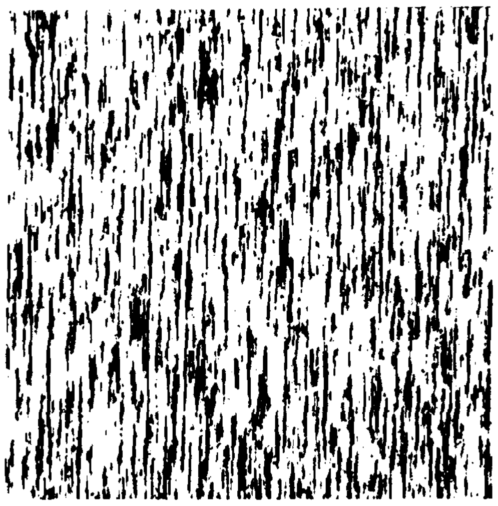

(c) Linear Model Simulation

Figure 3.11 Sand


(a) Original Texture


(b) N-gram Simulation

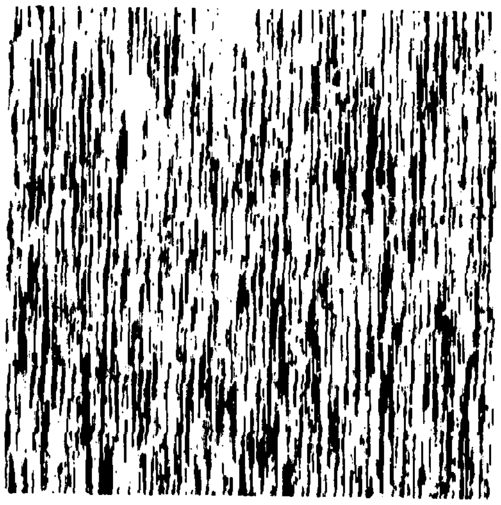
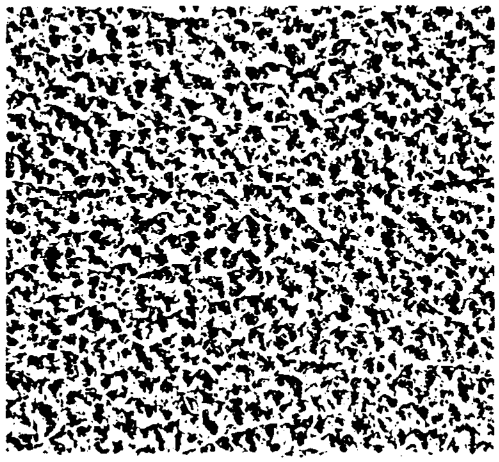
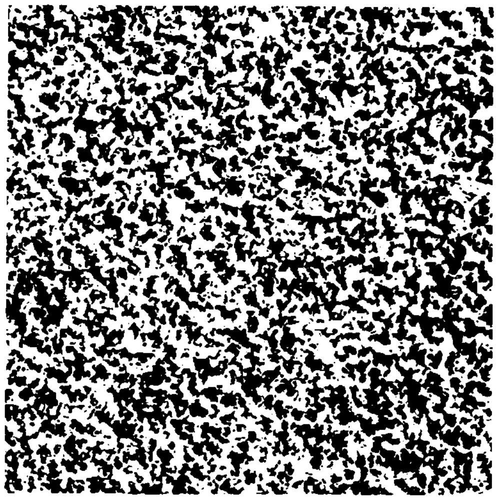

(c) Linear Model Simulation

Figure 3.12 Wood


(a) Original Texture


(b) N-gram Simulation

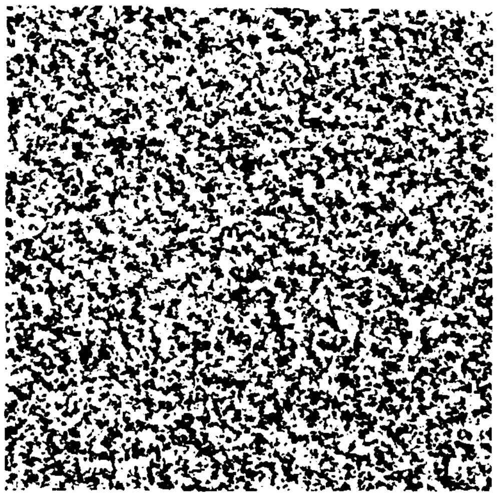
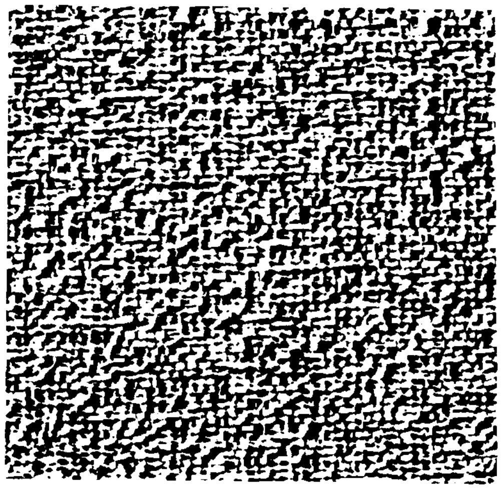
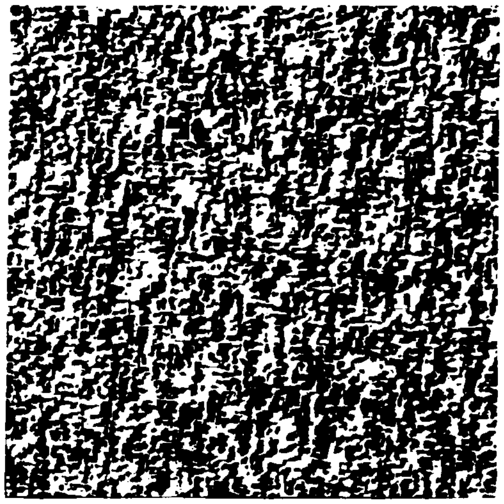

(c) Linear Model Simulation

Figure 3.13 Pigskin


(a) Original Texture


(b) N-gram Simulation

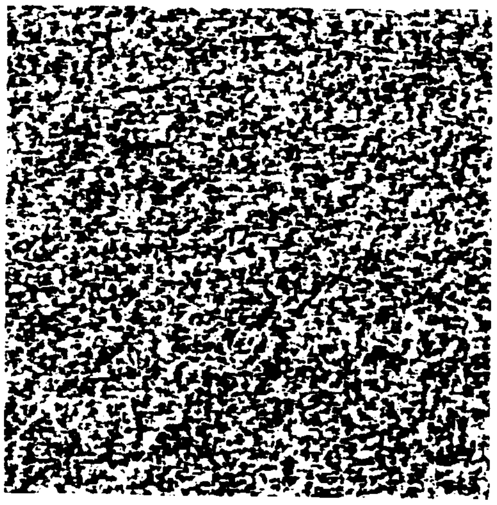
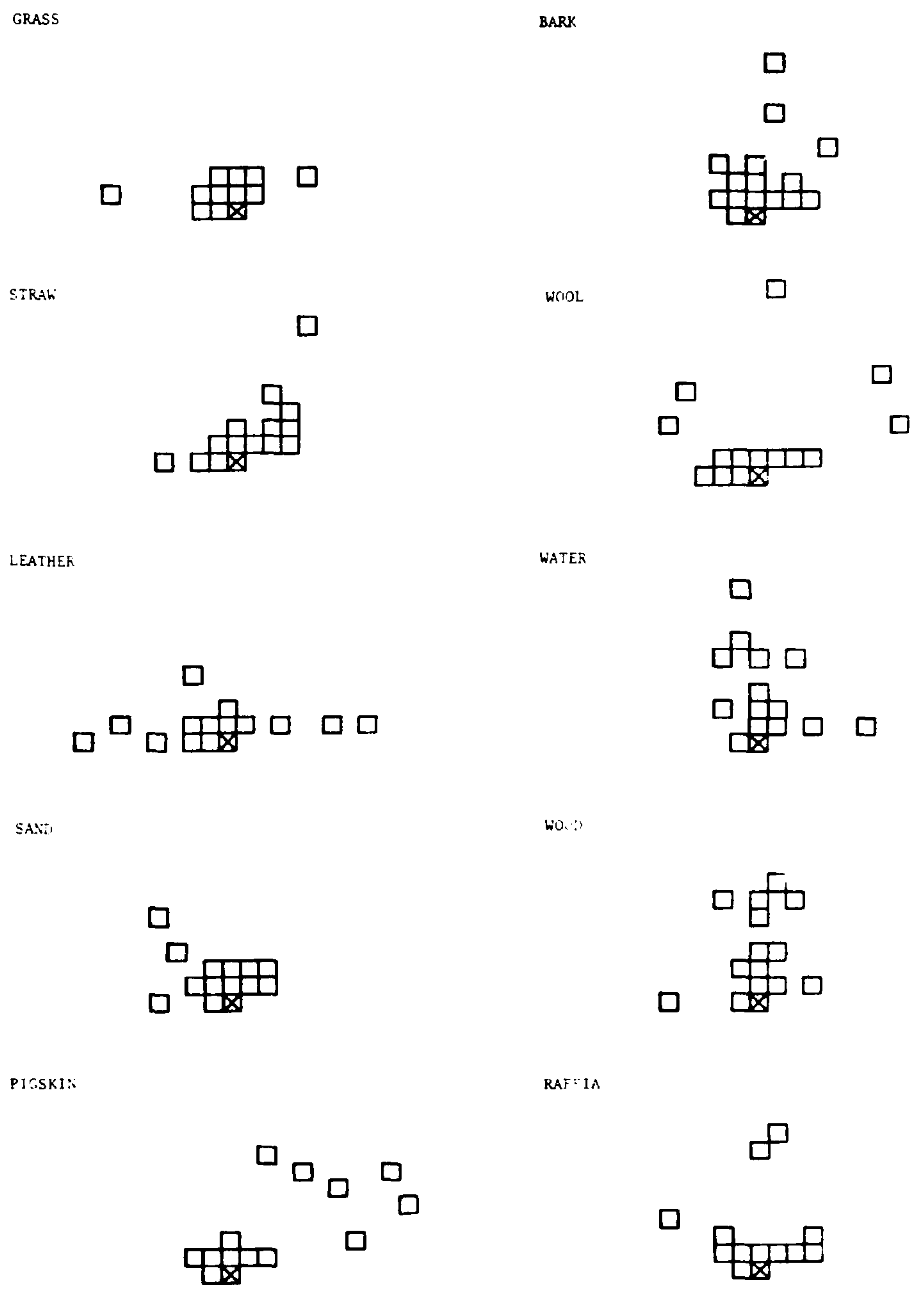

(c) Linear Model Simulation

Figure 3.14 Raffia

Table 3.1 TWO-DIMENSIONAL N-GRAM TEXTURE SIMULATION KERNELS



approach is not always evident. Some kernels, such as those for wool and water, seem to have a vertical or horizontal structure possibly resulting from the structure of their corresponding textures while others, such as the kernels for wood and raffia, do not. Still, the non-contiguous kernels produce excellent results.

3.9 Extension of N-Gram Model

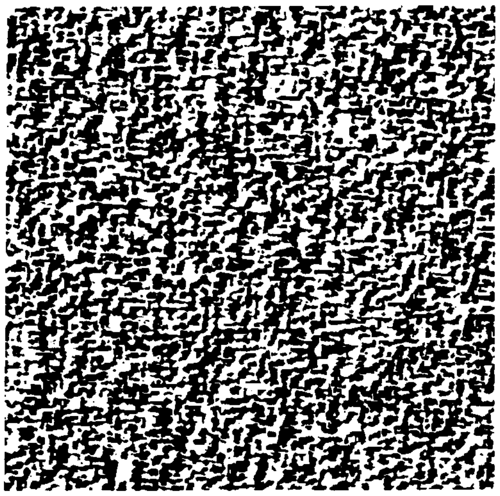
The N-gram model may be extended beyond the fixed N set by limited processor storage. The requirement that G must be estimated based on no fewer than q (which was set to 10 in our study) samples already reduces the storage to a maximum of M/q non-redundant parameters where M is the total number of samples in the parent texture image. For a 512x512 image, M is approximately 2.5×10^5. Given q = 10, this implies that a maximum of 25,000 generation parameters estimates must be stored. By increasing q, further reduction is possible. Meanwhile N is permitted to increase without bound to sample size limitations. For frequently occurring patterns, larger N's allow increasing dependency over larger distances. This is most desirable in regular and coarse-structured textures.

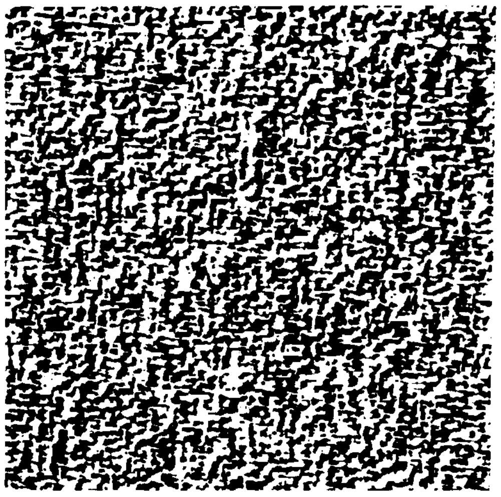
Searching for the proper generation parameter at each step in this type of process is complex. In the earlier N-gram storage, each pattern of 0's and 1's actually forms a binary address to the location in memory of the desired G. In the extended case a sort, search, or a series of comparisons along with some intelligent preprocessing is required. Efficiency is reduced.

To illustrate the effect of model extension the texture raffia was used. Figure 3.14(a) shows the original and Fig. 3.14(b) is the synthesis obtained using a kernel containing N = 14 pixels. Figure 3.14(c) is the linear model synthesis. Figures 3.15(a),(b),(c) were obtained using three different texture kernels with N = 22 points. Far more structure in these extended model versions is apparent. This is expected as at each generation step, the next pixel is allowed to depend on pixels further from it. As the pixels in the kernel become more widely spaced the synthesis becomes more structured but small, local regions often become more distorted and less raffia-looking because the information used in the synthesis process is more global than local.

3.10 Linear Model Generation of Binary Textures

The process of choosing the V_i's to be present in the texture generation kernel described in earlier sections of this chapter actually yields a simple linear model which can also be used to generate binary textures. The model which results from the determination of the generation


(a) N-gram Extended Model


(b) N-gram Extended Model

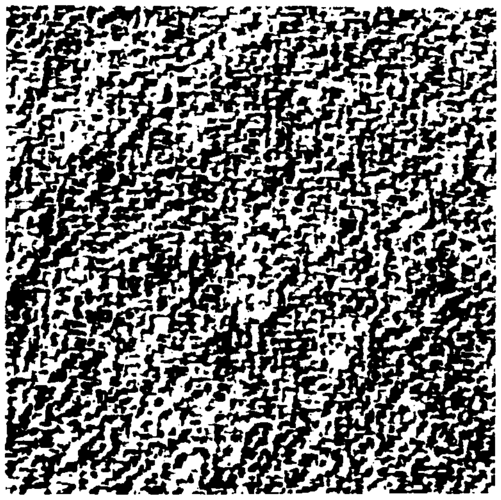

(c) N-gram Extended Model

Figure 3.15 Raffia

kernel may be expressed in equation form as

$$V_{N+1,k} = \beta_1 V_{1,k} + \beta_2 V_{2,k} + \ldots + \beta_N V_{N,k} + \beta_0 + \varepsilon_k \qquad (3.21)$$

or more simply as

$$V_{N+1} = \beta_1 V_1 + \beta_2 V_2 + \ldots + \beta_N V_N + \beta_0 + \varepsilon \quad . \qquad (3.22)$$

Once the estimates of the β_i's are known, a pixel V_{N+1} may be calculated from a set of given values V_i plus an error ε. In one-dimensional analysis this is sometimes known as the autoregressive time series model [24]. For binary V_i a value of V_{N+1} will be produced which is non-binary. To generate binary data using this model will therefore require quantization.

In the N-gram approach to texture simulation, the randomness of the texture is induced by the generation of a uniformly-distributed pseudo-random variable during the generation process. The comparison of this value with the estimate of the generation parameter, $\hat{G}$, yields the next binary pixel. A similar type of randomness must occur in the generation of binary textures using the linear model of Eq. (3.22). This randomness is expressed in the model in the error term ε.

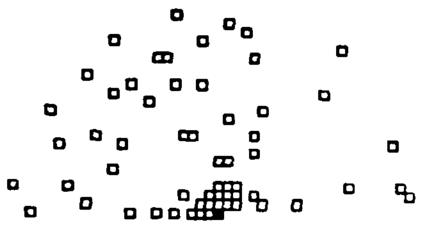
We can obtain an estimate of the distribution of ε in the same manner as we estimate the β's of the model. This may be done by applying the model to the sample data from which it was derived and observing the errors. That is, the linear model kernel is passed over the parent texture image and at each point a $\hat{V}_{N+1}$ is calculated based on Eq. (3.22) without the error term. Then $V_{N+1}-\hat{V}_{N+1}$ is calculated where V_{N+1} is the actual value of V_{N+1} in the parent texture. The histogram of the values can be used to estimate the distribution of ε. As one step further we could assume that has some known distribution such as Gaussian or normal, and merely estimate the parameters necessary to define this distribution. In the normal distribution case, only the standard deviation (or variance) of needs to be estimated. The mean of ε is zero in the linear model, least-squares distribution.

Our generation process then consists of the calculation of $\sum \beta_i V_i + \beta_0$ to which we add a random, normally-distributed error term ε and this value is then quantized to 0 or 1 based on comparison with 0.5. Results using this generation method are shown in Figs. 3.5(c) through 3.14(c). In these figures, N was allowed to be as large as 70 as only N coefficients (not 2^N) need to be stored along with $\hat{\sigma}_\varepsilon$, the estimate of the error standard deviation.

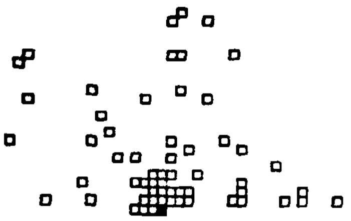
The kernels used in the linear model simulations are shown in Table 3.2. The linear model simulations are slightly inferior to the N-gram simulations but the degradation is far less than we would expect from such a massive compression of information (which is approximately 2 to 70). The results were good enough to encourage the application of the linear model to continuous-tone textures.

Table 3.2 TWO-DIMENSIONAL BINARY LINEAR MODEL TEXTURE GENERATION KERNELS

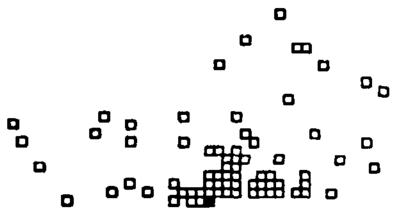
GRASS



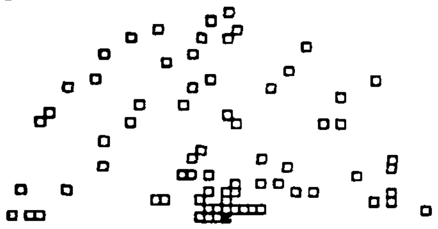
BARK



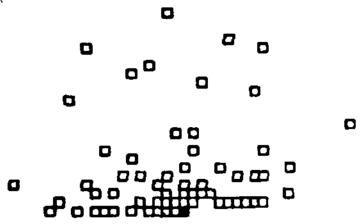
STRAW



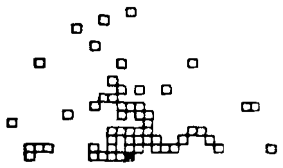
WOOL



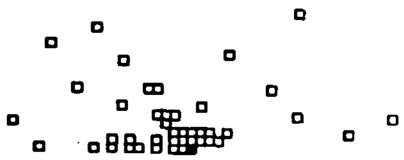
LEATHER



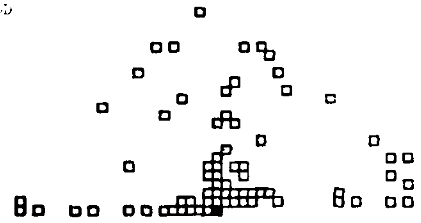
WATER



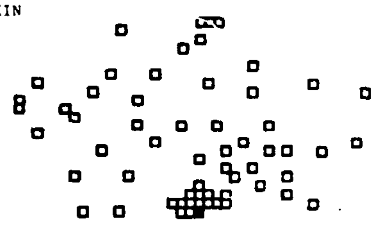
SAND



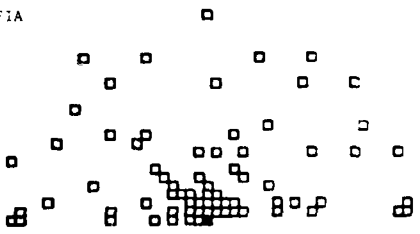
WOOD



PIGSKIN



RAFFIA



CHAPTER 4

ALGEBRAIC RECONSTRUCTION TECHNIQUE MODEL

4.1 Introduction

Julesz's conjecture [8] that second-order statistics are sufficient for human visual texture discrimination provides a useful estimate of the amount of information necessary to reconstruct a texture field. Although counterexamples to that conjecture have recently been found [4,25], and are shown in Chapter 2, it is a good first-order approximation. Examples of the use of that upper bound for texture analysis can be found in [14,26]. Therefore it is very tempting to use it for synthesizing natural texture fields. That is, we may attempt to simulate textures based on generation parameters estimated from 2-gram statistics over various distances. This approach requires fewer statistics to be collected from a parent texture as only 2-grams versus N-grams are collected.

This chapter illustrates that we must "invent" higher-order statistics to use the Markov generation coefficient approach for texture synthesis if we limit our knowledge of the original random field to second-order statistics. We will demonstrate how this can be done using Algebraic Reconstruction Techniques (ART).

4.2 Problem Definition

The stochastic approach toward texture analysis considers texture fields as samples of 2-D stochastic fields. Second-order statistics are given by the set of second-order joint density functions (2-grams), $P(V_i,V_j)$. As in Chapter 3, we also assume for simplicity that the random field is homogeneous, that is, invariant through translations. Given the set of joint density functions, $(P(V_i,V_j)$, for all spatial relationships between V_i and V_j, our problem is to synthesize a texture field with the same second-order statistics.

There are many ways of carrying out such a synthesis. We use a television raster type of scanning (left to right, top to bottom) and the generation kernel of Fig. 3.1(b). Even if we assume finite memory, namely that intensity at point V_{N+1} depends only on intensities at points located in some finite neighborhood, we see that second-order statistics are not sufficient to generate the process. Indeed, assuming a memory of order N, the intensity at V_{N+1} is computed using the conditional probability or generation coefficient

$$G_{V_{N+1}}(V_1,V_2,\dots,V_N) = P(V_{N+1}/V_1,V_2,\dots,V_N) \tag{4.1}$$

which involves (N+1)th-order joint density functions. Therefore if N is larger than 1 and we are given only second-order statistics, $P(V_i,V_j)$, we have to "invent" higher-order densities $P(V_1,V_2,\ldots,V_{N+1})$. Mathematically, the problem can then be stated as follows: given N random variables $V_1,\ldots,V_N$ such that for every pair (V_1,V_N), $1 \leq i,j \leq N$ and $i \neq j$, we know the joint density function $P(V_i,V_j)$, find a function $P(V_1,\ldots,V_N)$ which satisfies

$$P(V_1,V_2,\ldots,V_N) \geq 0 \quad \text{for all } V_1,\ldots,V_N \tag{4.2}$$

$$\sum_{V_1}\cdots\sum_{V_{i-1}}\sum_{V_{i+1}}\cdots\sum_{V_{j-1}}\sum_{V_{j+1}}\cdots\sum_{V_N} P(V_1,\ldots,V_i,\ldots,V_j,\ldots,V_N) = P(V_i,V_j) \quad . \tag{4.3}$$

Here the $P(V_i,V_j)$'s are sometimes called the marginals of $P(V_1,\ldots,V_N)$. Assuming quantization with g levels, the g^N unknowns $P(V_1,\ldots,V_N)$ can be stacked as a vector $\vec{p}$ and conditions (4.2) and (4.3) correspond to a linear programming problem:

$$\begin{cases} A\vec{p} = \vec{m} & (4.4) \\ \\ \vec{p} \geq 0 & (4.5) \end{cases}$$

where vector $\vec{m}$ is obtained from the functions $P(V_i,V_j)$ and matrix A of size $(\binom{M}{2}g^2)\mathrm{x}g^N$ contains only ones and zeroes. For any reasonable values of N and g, this is a set of linear equations and inequalities of fairly large dimension and the usual solution techniques such as the simplex method [4] become very limited.

4.3 Solution Through Algebraic Reconstruction Techniques (ART)

The ART algorithm was introduced by Gordon, Bender and Herman [27] for solving the problem of three-dimensional reconstruction from projections in electron microscopy and radiology. This is a deconvolution problem in which a function in a higher-dimensional space is estimated from its experimentally measured projections in a lower-dimensional space. For an excellent review of those techniques see Gordon [28].

The problem stated in Eqs. (4.2) and (4.3) or (4.4) and (4.5) is precisely of this form, where the projections are the second-order joint density functions. ART is therefore directly applicable. The simple intuitive interpretation is that each projected density is thrown back across the higher-dimensional region from whence it came, with repeated corrections to bring each projection of the estimate into agreement with the corresponding measured projection.

Formally, we use an iterative scheme defined by

$$\tilde{P}^{(q+1)}(V_1,\ldots,V_N) = \tilde{P}^{(q)}(V_1,\ldots,V_N)+tc^{(q)}(V_1,\ldots,V_N) \tag{4.6}$$

For all values of $V_1,\ldots,V_N$, for $q = 0,1,\ldots$

where the correction term $c^{(q)}(V_1,\ldots,V_N)$ is given by

$$c^{(q)}(V_1,\ldots,V_N) = \frac{1}{g^{N-2}\binom{N}{2}} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (\hat{P}(V_i,V_j)-\tilde{P}(V_i,V_j)) \tag{4.7}$$

and $\hat{P}(V_i,V_j)$ is the actual marginal measured, for example, from an original texture field. Here $\tilde{P}^{(q)}(V_i,V_j)$ is the marginal corresponding to the reconstructed density at iteration q.

We may express this in words as follows. The iterative process may be started with all reconstruction elements set to a constant $(\tilde{P}^{(0)}(V_1,\ldots,V_N) = 1/g^N$ for all $(V_1,\ldots,V_N)$. In each iteration the sum of the differences between the actual and the reconstructed marginals is computed and evenly divided amongst the g^{N-2} reconstruction elements. If the correction is negative, it may happen that the calculated density becomes negative at some points. This problem can be alleviated by using a modified iteration scheme defined by

$$\tilde{P}^{(q+1)}(V_1,\ldots,V_N) = \mathrm{MAX}\left\{0,\tilde{P}^{(q)}(V_1,\ldots,V_N)+tc^{(q)}(V_1,\ldots,V_N)\right\} \tag{4.8}$$

therefore guaranteeing $\tilde{P}^{(q+1)} \geq 0$ (constrained ART [28]). It is of course necessary to determine when an iterative algorithm has converged to a solution which is optimal according to some criterion. This is in turn related to the problem of finding the optimal value t_q of t. Various criteria for convergence have been devised [28]. For simplicity, we chose the mean-square error

$$\varepsilon_q = ||\vec{e}^{(q)}||_2^2 = ||\vec{m}-\vec{\tilde{m}}^{(q)}||_2^2 \tag{4.9}$$

between the measured and calculated marginals where $||\cdot||_2$ is the usual euclidean norm and $\vec{m}$ is defined in Eq. (4.4).

To derive the optimal step size, t, for each iteration we rewrite Eq. (4.6) in vector form as

$$\vec{\tilde{P}}^{(q+1)} = \vec{\tilde{P}}^{(q)} + t\vec{c}^{(q)} \quad . \tag{4.10}$$

Multiplying both sides of Eqs. (4.10) with matrix A (Eq. (4.4) we obtain

$$\vec{\tilde{m}}^{(q+1)} = \vec{\tilde{m}}^{(q)} + t\vec{d}^{(q)} \tag{4.11}$$

where

$$\vec{d}^{(q)} = A\ \vec{c}^{(q)} \tag{4.12}$$

and subtracting the actual marginal vector $\vec{\tilde{m}}$ from both sides of Eq. (4.11) yields

$$\begin{aligned} ||\vec{e}^{(q+1)}||_2^2 &= ||\vec{e}^{(q)} - t\vec{d}^{(q)}||_2^2 \\ &= t^2||\vec{d}^{(q)}||_2^2 - 2t\vec{d}^{(q)}\cdot\vec{e}^{(q)} + ||\vec{e}^{(q)}||_2^2 \ . \end{aligned} \tag{4.13}$$

Therefore the error ε_{q+1} at iteration q+1 is minimized for

$$t_q = \frac{\vec{e}^{(q)} \cdot \vec{d}^{(q)}}{||\vec{d}^{(q)}||_2^2} \tag{4.14}$$

where • denotes the inner product.

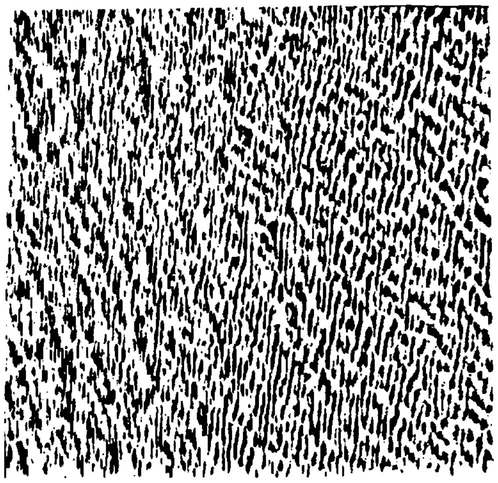
A dual approach also explored is based upon the analysis of Eq. (4.3) in the Fourier domain. It can easily be shown that the initial problem stated by Eqs. (4.2) and (4.3) is equivalent to an interpolation problem in the Fourier domain. The major drawback of this approach is the difficulty to ensure the positivity of the inverse Fourier transform of the interpolated function. Therefore this method was not pursued even though it may be the case that "good" interpolating functions will alleviate that problem.

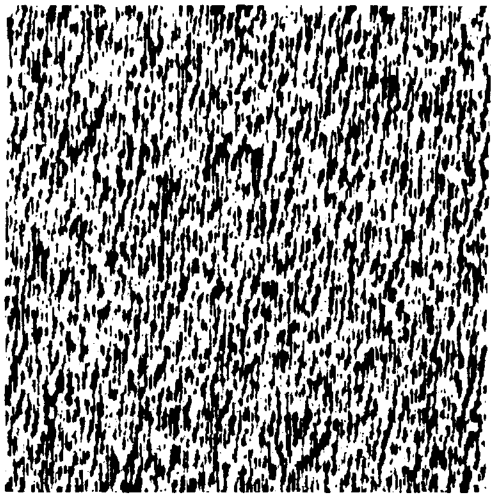
The basic philosophy of the two approaches just discussed is that Nth-order joint density functions are "invented" to satisfy exactly the constraints stated in Eq. (4.3). Their obvious disadvantage is the high dimensionality (g^N for an Nth-order joint density function) of the data that is to be stored compared with the usually lower dimensionality ($\binom{N}{1} g^2$ for the second-order joint density functions) of the data that is effectively used.

4.4 Results and Conclusions

The iterative process defined in Eq. (4.6) may be halted at any number of iterations, q, and a texture may be generated using the value of $\vec{\tilde{p}}$ at that point. However, it should be kept in mind that the success of a texture synthesis depends on making the error $c^{(q)}$ as small as possible and that the texture generation process is sensitive to this error. It has also been found by experimentation that the $\vec{\tilde{p}}$ contains many values which are set to zero by implementation of constrained ART. This tends to cause the Markov texture generating process to become absorbing, which causes patches of white and black or streaks and lines to be generated. This is eliminated by setting those values which are zero to a small non-zero value, δ , in the generation process.

Using the above concepts, texture simulations of the binary textures water (Fig. 4.1(a)) and raffia (Fig. 4.2(a)) were generated (Figs. 4.1(b),4.2(b)). Textures similar to those in Chapter 3 employing actual Nth-order statistics (N=14) were also generated (Figs. 4.1(c),4.2(c)) and are included here for reference. The V_i in the generation kernels for the ART model and the N-gram model were chosen using the ideas described earlier in Chapter 3.


(a) Original Texture


(b) ART-generated Simulation

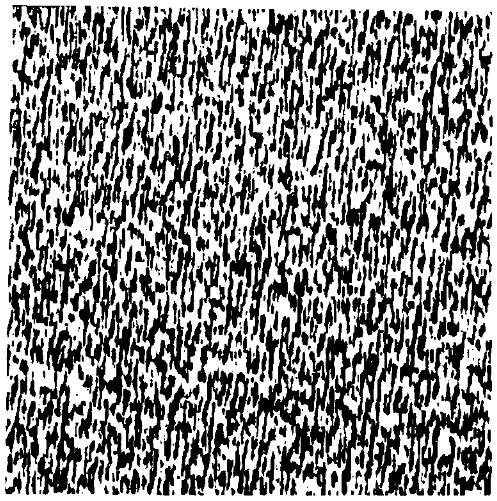
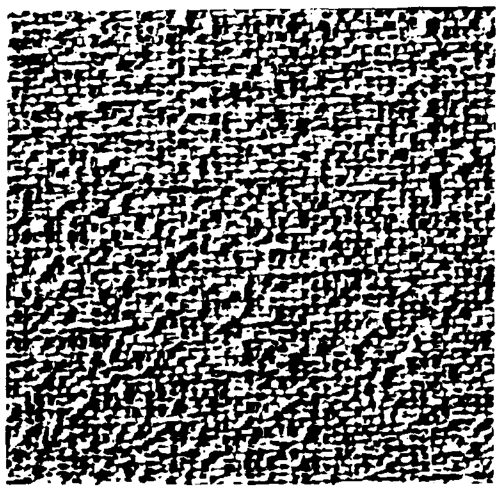
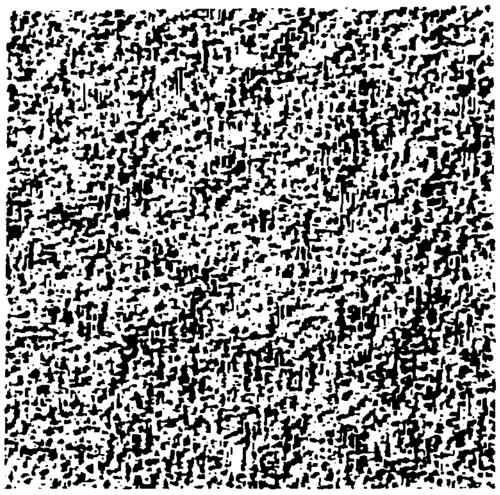

(c) N-gram Simulation

Figure 4.1 Water


(a) Original Texture


(b) ART-generated Simulation

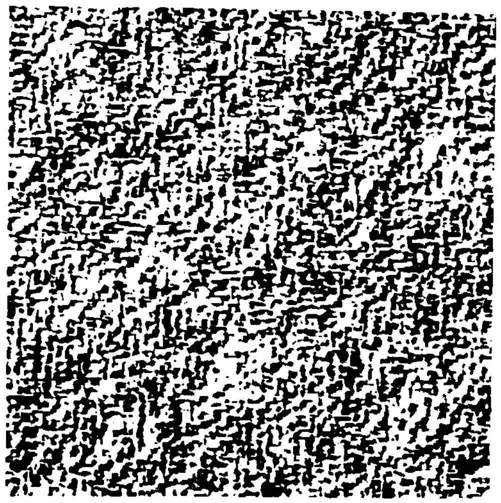

(c) N-gram Simulation

Figure 4.2 Raffia

At the onset of investigation of the Algebraic Reconstruction Model it was hoped that this approach would be useful for generating continuous-tone textures. However, using it to generate binary textures revealed that the convergence for the iterative process was very slow even with the optimal step size t. Each iteration required much computation and the storage required for the N-grams was large. More than eight hours of CPU time on a DEC KL-10 were required to execute the large number of iterations required for a visually pleasing solution. Generating textures based on estimated N-grams which is detailed in Chapter 3 is probably more efficient and less complex computationally. This work did lead to other texture simulation models (see Faugeras [29]).

The results using algebraic reconstruction are nearly equivalent to the results using complete N-grams as second-order statistics collected from binary textures contain a great deal of information. Close examination of the textures generated using algebraic reconstruction on a high-resolution display device reveals a high random noise level. The computational requirements and the final noise level indicate that the N-gram method of generating textures is much less complex and yields better visual results than the method employing algebraic reconstruction.

CHAPTER 5

CONTINUOUS-TONE LINEAR TEXTURE MODEL

5.1 Introduction

In this chapter, the concept of using a linear model for generating binary textures that was briefly discussed in Chapter 3 is extended to multi-grey-level (256-level) or continuous-tone textures. Pictorial results of the application of this model to simulation of a variety of textures are presented.

The application of the linear or autoregressive model to time series processes has been extensive. These applications, which range from weather forecasting to stock market predictions, primarily utilize the models and concepts introduced by Box and Jenkins [24]. Many authors have expanded and elaborated on their approaches [30-38,62]. Some researchers have applied these ideas to texture simulation, in fact, the Box-Jenkins autoregressive model is one of the very few approaches to texture synthesis presented thus far.

McCormick and Jayaramamurthy [2] were perhaps the first to make a notable attempt to simulate natural textures using this approach. Their work consisted of a discussion of the Box and Jenkins autoregressive (AR), moving average (MA) and autoregressive integrated moving average (ARIMA) models including estimation of model parameters and adequacy of model fit. (These terms are later defined in this chapter.) A very simple model was then used to simulate two very similar textures which closely resemble the wood texture of this study by filling in the holes of a parent texture using the derived model. Only two textures, both exhibiting a wood-grain-like structure, were used. Similar work was done later by Tou, Kao and Chang [11]. Unfortunately, the results of their simulation of these textures were displayed using a printout of Chinese characters and so the degree of success of their method is unclear. The appearance of texture synthesis results on a computer printout will confuse most observers unaccustomed to such crude image displays. The models were again very simple and contained no more than three terms in the linear model summation. Deguchi and Morishita [12] attempted to use the linear model to segment and partition textures. Their approach was only partially successful.

In the above simulation attempts, the models used were simple. The process of collecting statistics and estimating parameters is complex. In some cases, previous authors attempted to use the complex Box and Jenkins ARIMA model which leads to difficult model parameter estimation if the number of model elements is greater than two or three.

In our study, the simpler autoregressive model is used and is allowed to contain a large number of parameters. This is possible using the assumption of homogeneity (stationarity) combined with the forward selection process of choosing non-contiguous generation kernels as described in Chapter 3. These models are extended further by allowing second-order autoregressive models and non-stationary noise. Results of texture simulations using these models are included in this chapter.

5.2 The Linear Autoregressive Model

In Chapter 3 the linear autoregressive model, used to determine the elements of the generating kernel, was expressed as

$$Y_k = \vec{X}_k^T \vec{\beta} + \varepsilon_k \qquad k = 1,\ldots,M \tag{5.1}$$

where

$$Y_k = V_{N+1,k}$$

and

$$\vec{X}_k = \begin{bmatrix} 1 \\ V_{1,k} \\ V_{2,k} \\ \vdots \\ V_{N,k} \end{bmatrix} .$$

Here $\vec{\beta}$ is an (N+1)x1 vector of unobservable parameters and ε_k is an unobservable random variable such that $E[\varepsilon_k] = 0$. The sample number (index) is denoted by k and M is the total number of observations. We can also define the vectors $\vec{Y}$ and $\vec{\varepsilon}$ and the matrix X by

$$X = \begin{bmatrix} \vec{X}_1^T \\ \vec{X}_2^T \\ \cdot \\ \cdot \\ \cdot \\ \vec{X}_M^T \end{bmatrix} \qquad \vec{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \cdot \\ \cdot \\ \cdot \\ Y_M \end{bmatrix} \qquad \vec{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon_M \end{bmatrix} \tag{5.2}$$

and our model may be expressed as

$$\vec{Y} = X\vec{\beta} + \vec{\varepsilon} \quad . \tag{5.3}$$

In equation form, dropping the k subscript, the model becomes

$$V_{N+1} = \beta_1 V_1 + \beta_2 V_2 + \ldots + \beta_N V_N + \beta_0 + \varepsilon . \qquad (5.4)$$

Sums and sums of squares leading to the calculation of the correlation or covariance matrix of the parent texture are obtained by passing any chosen generation kernel pattern over the texture. From this matrix, the least-squares parameter estimate of $\vec{\beta}$ is obtained. The multiple-pass forward selection process described in Chapter 3 leads to a final linear autoregressive model which is then used to generate textures.

5.3 Autoregressive and Moving Average Models

In this section, we will introduce the general linear model as defined by Box and Jenkins [24] and Grabill [39] and discuss the relationships that exist between it and the autoregressive and moving average models. When the autoregressive model is extended without bound it is essentially equivalent to the general linear model and the moving average model is a subset of general linear models. Allowing for a large model reduces the complexity of parameter estimation and allows easy selection of a generating kernel and model.

Many of the equations in this section also assume that the texture is one-dimensional and that the generating kernel is contiguous. That is, V_{N+1} follows

V_N. This is consistent with notation presented earlier if images are expressed in lexicographic [40,41] notation as one-dimensional vectors. The non-contiguous kernel may be expressed as contiguous if the β_i are zero in the model.

The output of a linear filter whose input is white noise ε_k may be described using the general linear time series model

$$\begin{aligned}\tilde{V}_{N+1} &= \varepsilon_{N+1} + \psi_0\varepsilon_N + \psi_1\varepsilon_{N-1} + \psi_2\varepsilon_{N-2} + \cdots \\ &= \varepsilon_{N+1} + \sum_{j=1}^{\infty}\psi_j\varepsilon_{N-j}\end{aligned} \tag{5.5}$$

where $\tilde{V}_{N+1}=V_{N+1}-\mu$ and μ is the mean of the process, assumed to be stationary here. Thus $\tilde{V}_{N+1}$ is a weighted sum of present and past values of the white noise process ε_k . ε_k usually has zero mean and constant variance σ_ε^2. The auto-covariance is also defined as

$$\gamma_t = E[\varepsilon_k\varepsilon_{k+t}] = \begin{cases}\sigma_\varepsilon^2 \\ 0\end{cases} \quad . \tag{5.6}$$

Notice that Eq. (5.5) may be written in a different form

$$\tilde{\vartheta}_{N+1} = \pi_0\tilde{\vartheta}_N + \pi_1\tilde{\vartheta}_{N-1} + \pi_2\tilde{\vartheta}_{N-2} + \ldots + \varepsilon_{N+1}$$

$$= \sum_{j=0}^{\infty}\pi_j\tilde{\vartheta}_{N-j} + \varepsilon_{N+1} \quad . \tag{5.7}$$

Equation (5.7) may be derived from Eq. (5.5) as

$$\varepsilon_{N+1} = \tilde{\vartheta}_{N+1} - \sum_{j=0}^{\infty}\psi_j\varepsilon_{N-j} \tag{5.8}$$

$$\varepsilon_N = \tilde{\vartheta}_N - \sum_{j=1}^{\infty}\psi_j\varepsilon_{N-j} \quad . \tag{5.9}$$

Therefore

$$\tilde{\vartheta}_{N+1} = \varepsilon_{N+1} + \psi_0\left[\tilde{\vartheta}_N - \sum_{j=1}^{\infty}\psi_j\varepsilon_{N-j}\right] + \sum_{j=1}^{\infty}\psi_j\varepsilon_{N-j}$$

$$= \varepsilon_{N+1} + \psi_0\tilde{\vartheta}_N + (1-\psi_0)\sum_{j=1}^{\infty}\psi_j\varepsilon_{N-j} \quad . \tag{5.10}$$

Similarly

$$\varepsilon_{N-1} = \tilde{\vartheta}_{N-1} - \sum_{j=2}^{\infty}\psi_j\varepsilon_{N-j} \tag{5.11}$$

$$\Rightarrow \tilde{\vartheta}_{N+1} = \varepsilon_{N+1} + \psi_0\tilde{\vartheta}_N + (1-\psi_0)\psi_1\varepsilon_{N-1} + (1-\psi_0)\sum_{j=2}^{\infty}\psi_j\varepsilon_{N-j} \tag{5.12}$$

$$= \varepsilon_{N+1} + \psi_0\tilde{\vartheta}_N + (1-\psi_0)\psi_1\left[\tilde{\vartheta}_{N-1} - \sum_{j=2}^{\infty}\psi_j\varepsilon_{N-j}\right] + (1-\psi_0)\sum_{j=2}^{\infty}\psi_j\varepsilon_{N-j}$$

$$= \varepsilon_{N+1}+\psi_0\tilde{V}_N+(1-\psi_0)\psi_1\tilde{V}_{N-1}+(1-\psi_0)(1-\psi_1)\sum_{j=2}^{\infty}\psi_j\varepsilon_{N-j} \quad .$$

And so, by continuing this process Eq. (5.7) is found.

It may also be shown that Eq. (5.5) can be rewritten as

$$\tilde{V}_{N+1} = \psi(B)\varepsilon_{N+1} \tag{5.13}$$

where B is the backward shift operator

$$B\varepsilon_k = \varepsilon_{k-1} \tag{5.14}$$

$$B^j\varepsilon_k = \varepsilon_{k-j} \tag{5.15}$$

and

$$\psi(B) = \sum_{j=0}^{\infty}\psi_{j-1}B^j \tag{5.16}$$

where $\psi_{-1} = 1$. $\psi(B)$ is often called the transfer function of the linear filter.

As certain constants or parameters must be estimated from the sample data available it is sometimes important to minimize the number of parameters required to accurately represent a process. This simplifies analysis of a model and reduces the number of required computations. The general linear model containing an infini number of terms is of little practical value. Therefore, if is often expedient to allow the general linear time series process to be reduced to a model in which the current state of the process may be expressed as a finite aggregate of previous values of the process and a driving error value ε_{N+1}. This autoregressive (AR) model may be written as

$$\tilde{V}_{N+1} = \phi_0\tilde{V}_N + \phi_1\tilde{V}_{N-1} + \phi_2\tilde{V}_{N-2} + \ldots + \phi_p\tilde{V}_{N-p} + \varepsilon_{N-1} \quad . \tag{5.17}$$

We may define the autoregressive operator $\phi(B)$ as

$$\phi(B) = (1-\phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p) \tag{5.18}$$

and thus Eq. (5.17) may be rewritten as

$$\phi(B)\,\tilde{V}_{N+1} = \varepsilon_{N+1} \; . \tag{5.19}$$

The moving average (MA) model may be written as a special case of Eq. (5.5) where only the first g+1 of the ψ weights are non-zero. This process is

$$\begin{aligned}\tilde{V}_{N+1} &= \varepsilon_{N+1} - \theta_0\varepsilon_N - \theta_1\varepsilon_{N-1} - \cdots - \theta_p\varepsilon_{N-p} \\ &= \varepsilon_{N+1} - \sum_{j=0}^{q}\theta_j\varepsilon_{N-j} \quad .\end{aligned} \tag{5.20}$$

As in the case of the autoregressive model, we may write the moving average operator as

$$\theta(B) = (1-\theta_0 B-\theta_1 B^2- \ldots - \theta_{q-1}B^q-\theta_q B^{q+1}) \tag{5.21}$$

and thus Eq. (5.20) may be rewritten as

$$\tilde{V}_{N+1} = \theta(B)\varepsilon_{N+1} \tag{5.22}$$

A model incorporating the finite-term concept of both the autoregressive and moving average model may be written as

$$\begin{aligned}\tilde{V}_{N+1} &= \phi_0 V_N+\phi_1 \tilde{V}_{N-1}+\ldots+ \phi_p \tilde{V}_{N-p}+\varepsilon_{N-1} \\ &\quad - \theta_0\varepsilon_N- \theta_1\varepsilon_{N-1}- \ldots - \theta_q\varepsilon_{N-q}\end{aligned} \tag{5.23}$$

or

$$\phi(B)\tilde{V}_{N+1} = \theta(B)\varepsilon_{N+1} \quad . \tag{5.24}$$

This mixed autoregressive-moving average (ARMA) processing can be thought of as the output $\tilde{V}_{N+1}$ from a linear filter whose transfer function is the ratio of two polynomials $\theta(B)$ and $\phi(B)$ when the input is white noise ε_t.

An Example

To show the relationship between an autoregressive process and a moving process we will consider the simple example

$$\tilde{V}_{N+1} = \varepsilon_{N+1} - \theta_0 \varepsilon_N \qquad (5.25)$$

which is a simple moving average process. The above equation indicates that

$$\tilde{V}_N = \varepsilon_N - \theta_0 \varepsilon_{N-1} \qquad (5.26)$$

This implies that

$$\varepsilon_N = \tilde{V}_N + \theta_0 \varepsilon_{N-1} \qquad (5.27)$$

Substituting Eq. (5.27) into Eq. (5.25) yields

$$\tilde{V}_{N+1} = -\theta_0 \tilde{V}_N - \theta_0^2 \varepsilon_{N-1} + \varepsilon_{N+1} \qquad (5.28)$$

and by a similar substitution of

$$\varepsilon_{N-1} = \tilde{V}_{N-1} + \theta_0 \varepsilon_{N-2} \qquad (5.29)$$

into Eq. (5.28) we see that

$$\tilde{V}_{N+1} = -\theta_0 \tilde{V}_N - \theta_0^2 V_{N-1} - \theta_0^3 \varepsilon_{N-2} + \varepsilon_{N+1} \qquad (5.30)$$

Continuing this process yields

$$\tilde{V}_{N+1} = -\sum_{j=1}^{\infty} \theta_0^j V_{(N+1)-j} + \varepsilon_{N+1} \qquad (5.31)$$

Thus, a finite moving average process may be written as an infinite autoregressive process.

5.4 Parsimony Between Models

The above equations show how the general linear time series model is related to the autoregressive, moving average and autoregressive-moving average models. The concept of parsimony suggests that is is usually desirable to express a model with as few terms as possible. However, it has been shown that if we are willing to sacrifice the concept of parsimony and deal with infinite (or large) general linear models we can use the infinite autoregressive model (Eq. (5.7)), also known as the general linear model, to describe any general linear time series process.

In theory, it is possible to approximate as closely as desired any general linear time series process with a finite autoregressive process of order p by allowing p to increase until the desired closeness is obtained. In application, however, the estimation of model parameters π_i of Eq. (5.7) may be less accurate than desired due to the noise ε_k in the system. That is, the noise of a system will cause error in the estimation of the π_i to the extent that some π_i's are believed to be zero or are estimated so poorly that an inaccurate model is developed. It is not clear to what extent the accuracy of a model is improved in such a case by using an autoregressive moving average model. It is clear, however, that going to such a model causes increased complexity in the parameter estimation process and induces difficulty into the hypothesis testing process which is simple in an autoregressive model case. Parameter estimation for the ARMA model requires a multiple-pass, extensive and computationally-complex iterative process which is complicated by the required extension of our model in a two-dimensional image to many points and very large sample size. For this reason, the ARMA model was not used for texture simulation in this thesis.

5.5 Results

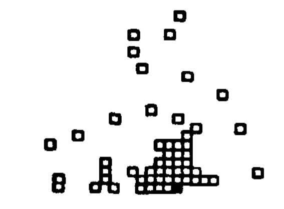
The linear (autoregressive) model of Eq. (5.4) was used to simulate a variety of natural textures. Stationary, independent Gaussian noise was used to drive the synthesis process. The variance of the noise was estimated by applying the model to the sample data and observing the prediction errors. Images resulting from the fitting of estimated models to sample data are shown later in this chapter. These errors, which are often called residuals, are pixels formed by the difference $\hat{V}_{N+1}-V_{N+1}$ where V_{N+1} is the actual observed pixel value of the sample parent image and $\hat{V}_{N+1}$ is the corresponding fitted value obtained by use of the linear model. The standard deviation of these errors can be measured and used as the standard deviation of pseudo-random normally-distributed noise in the generation process. Actually, this information can also be obtained during the decomposition of the covariance matrix.

The number of pixels in each generation kernel, N, varied from 30 to 60. The generation kernels used for each simulation are shown in Table 5.1. The simulation results are shown in Fig. 5.1(b) through Fig. 5.11(b).

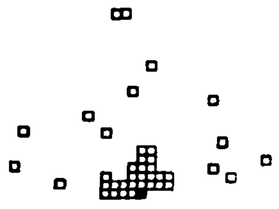
These simulations indicate that the linear model using stationary gaussian noise produces acceptable simulations of a variety of textures including grass, wool, leather, sand and water. As with the binary model, the simulation of bark (Fig. 5.2) shows absence of macro texture. The non-homogeneities present in the straw texture (Fig. 5.3) cause an "average" straw to be generated which is very similar to the binary texture simulation. The cloth texture (Fig. 5.4) is composed of two subtextures and therefore simulations made using statistics measured over the whole image will be a mixture of the two subtextures. The simulation of raffia (Fig. 5.11) has good structure as the linear model kernel is large but sharp edges present in the original texture are absent in the simulation. The same is true of the

Table 5.1 TWO-DIMENSIONAL LINEAR MODEL TEXTURE GENERATION KERNELS

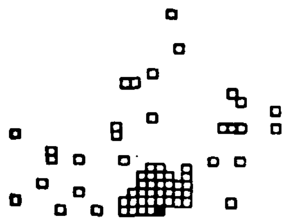
GRASS



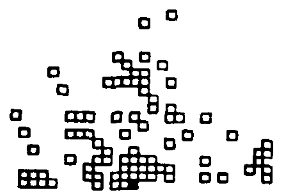
BARK



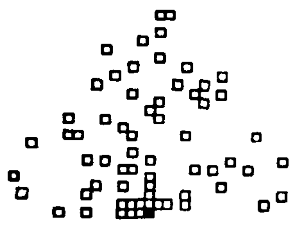
STRAW



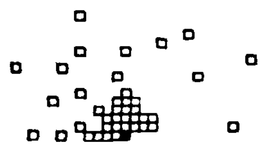
CLOTH



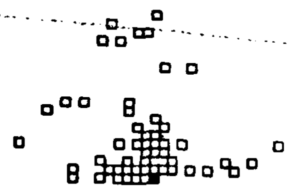
WOOL



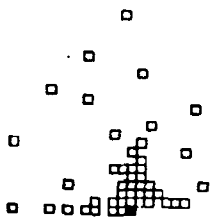
SAND



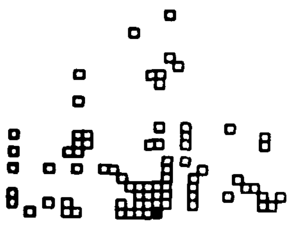
WATER



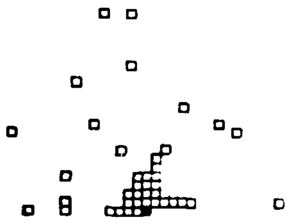
WOOD

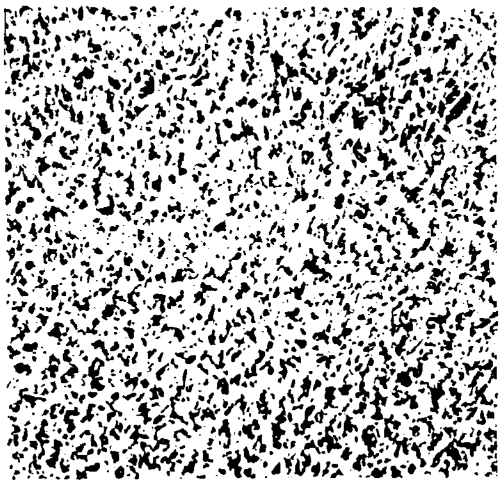


RAFFIA

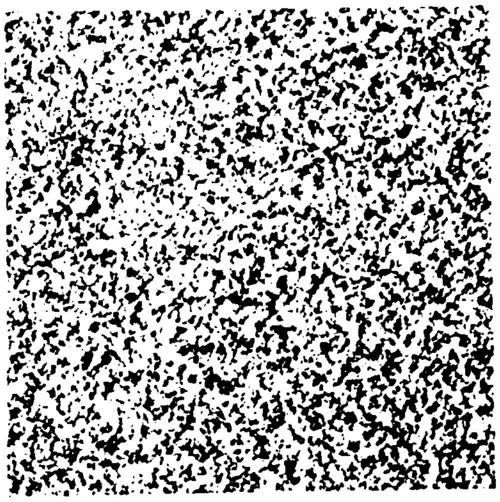


BUBBLES

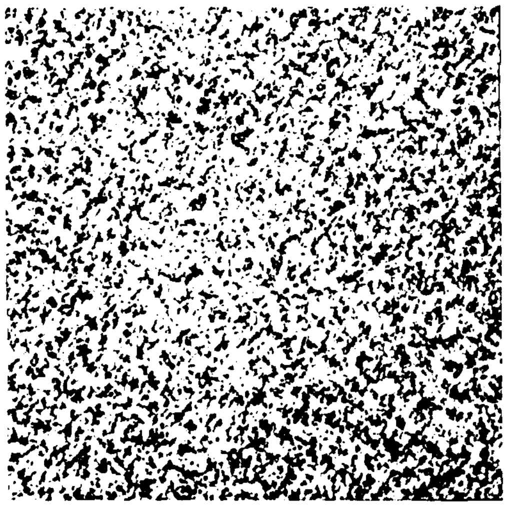




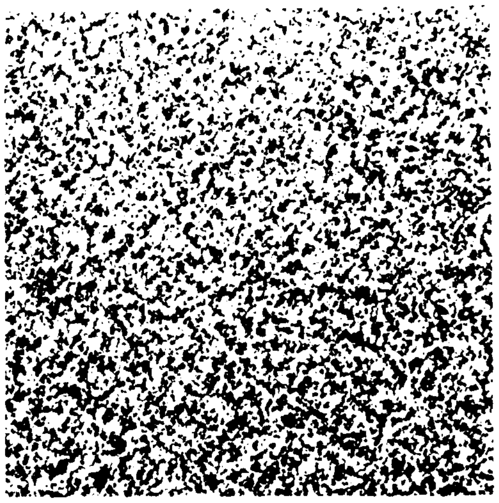
(a) Original Texture



(b) First-order Linear Model

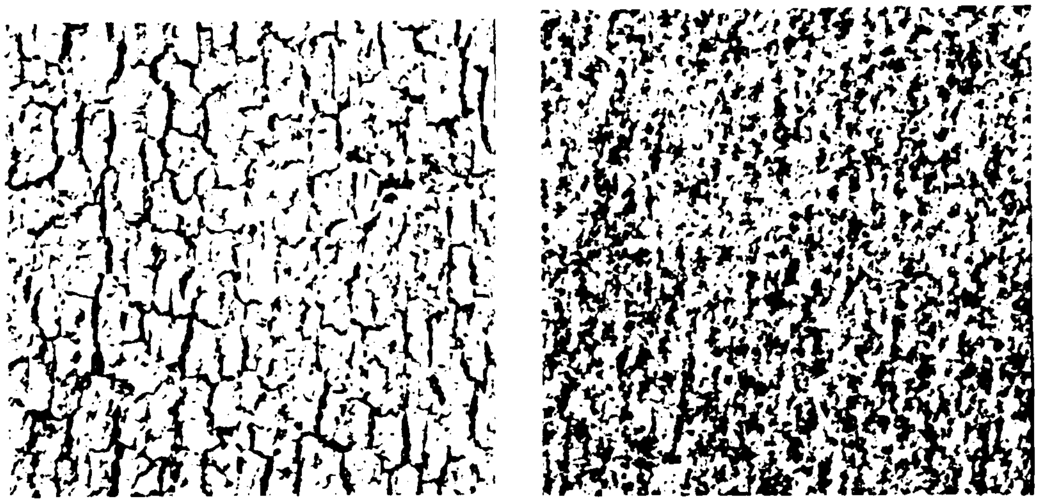


(c) Second-order Linear Model



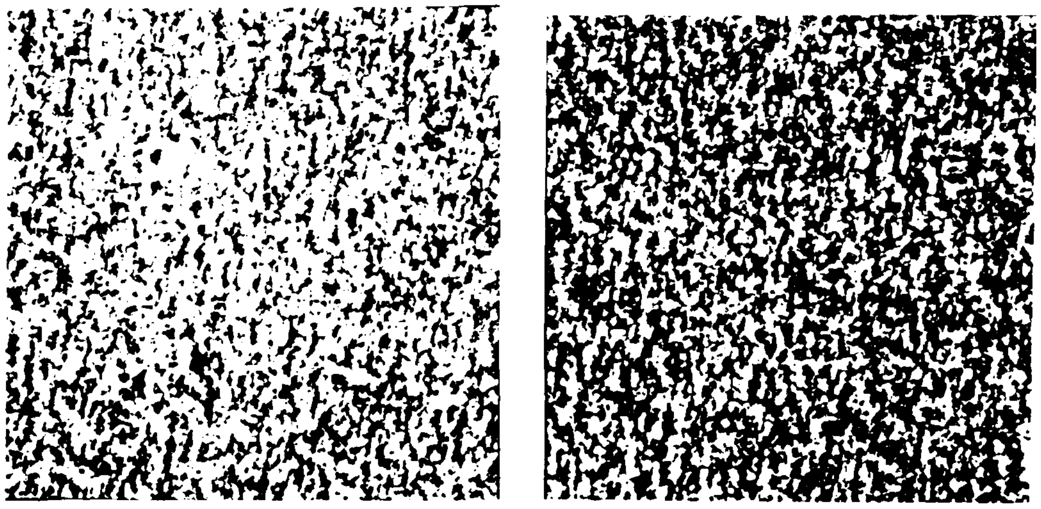
(d) Second-order Linear Model with Non-stationary Noise

Figure 5.1 Grass



(a) Original Texture

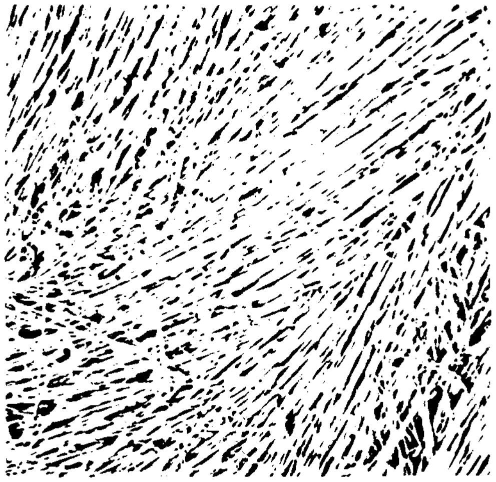
(b) First-order Linear Model

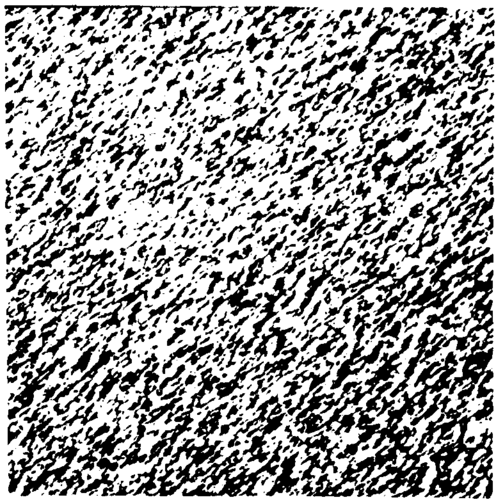


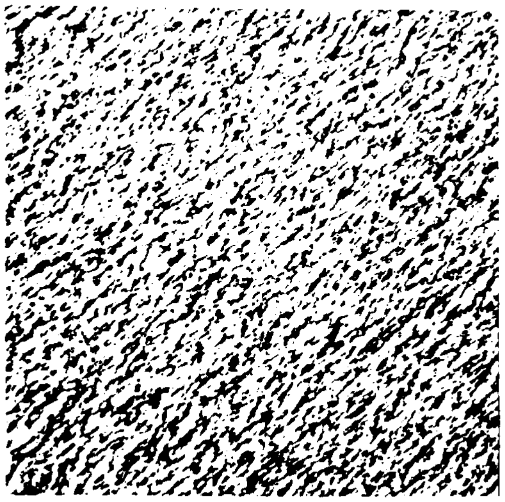
(c) Second-order Linear Model

(d) Second-order Linear Model with Non-stationary Noise

Figure 5.2 Bark


(a) Original Texture


(b) First-order Linear Model


(c) Second-order Linear Model

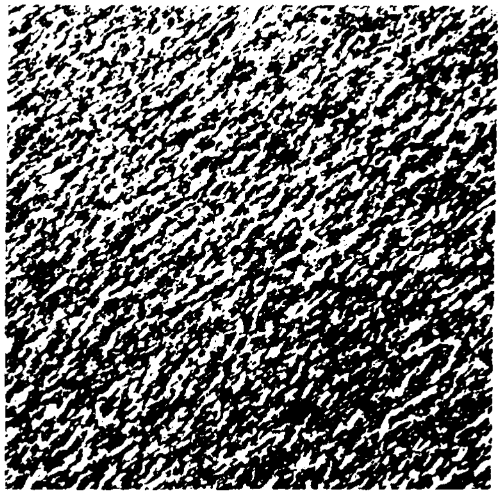
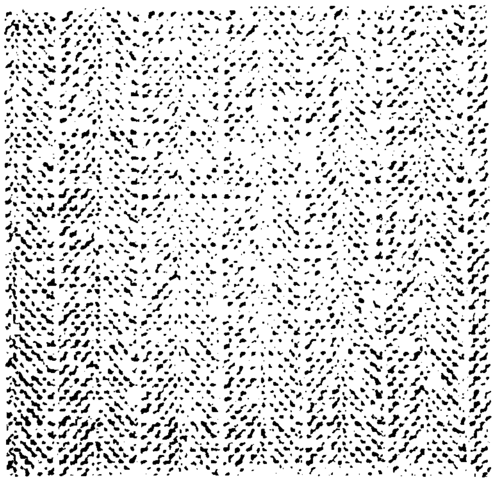
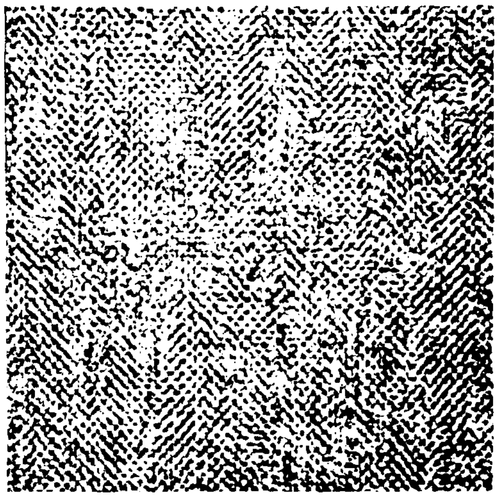
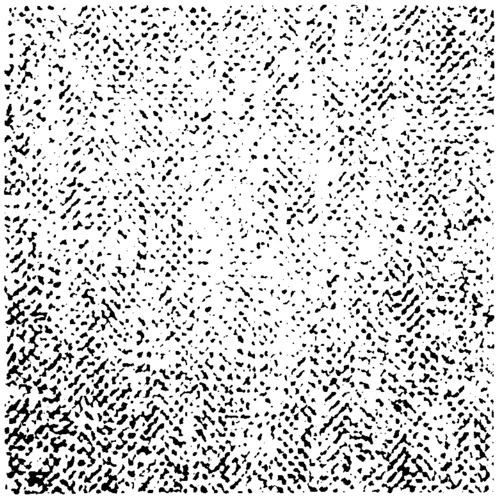

(d) Second-order Linear Model with Non-stationary Noise

Figure 5.3 Straw


(a) Original Texture


(b) First-order Linear Model


(c) Second-order Linear Model

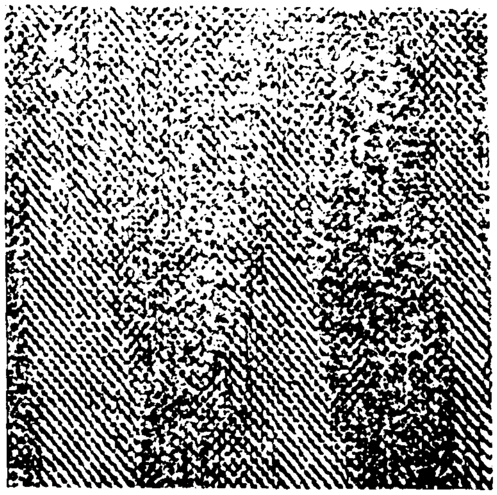
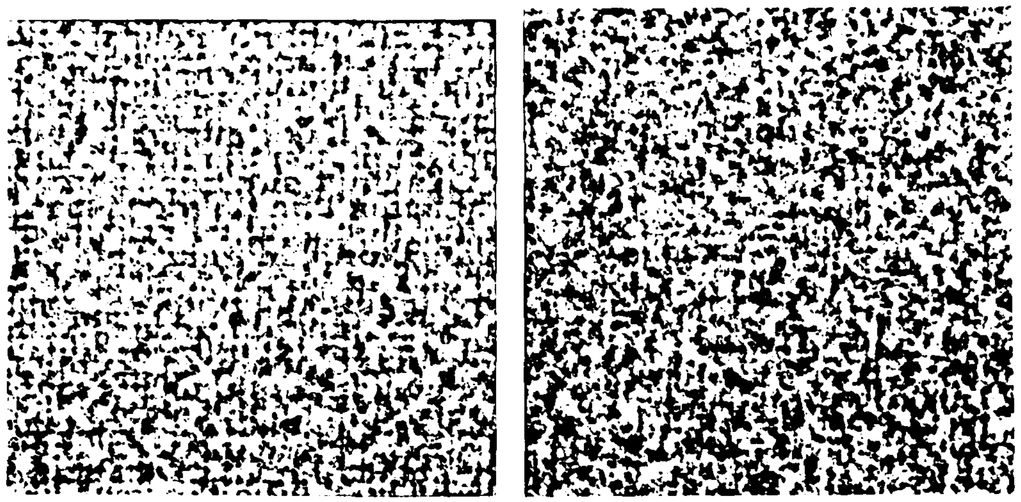
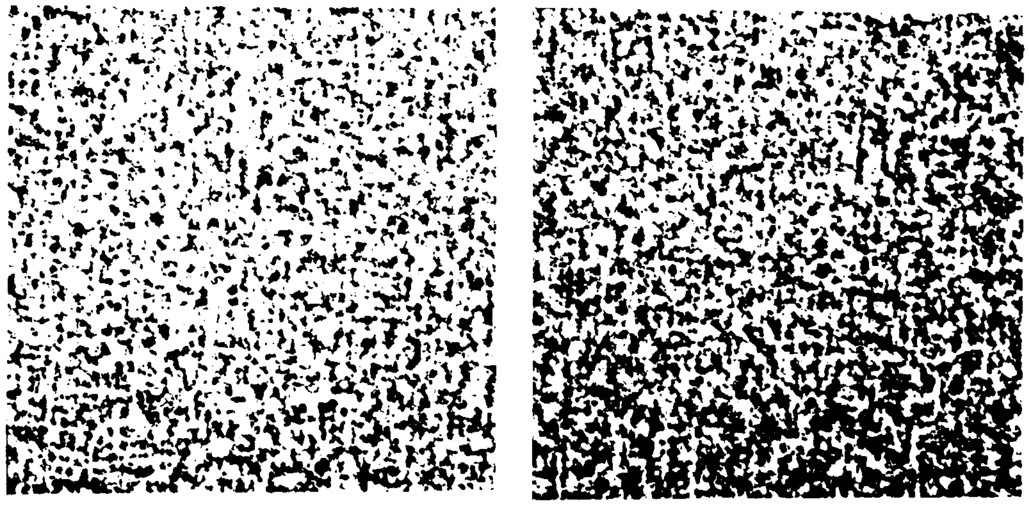

(d) Second-order Linear Model with Non-stationary Noise

Figure 5.4 Cloth



(a) Original Texture

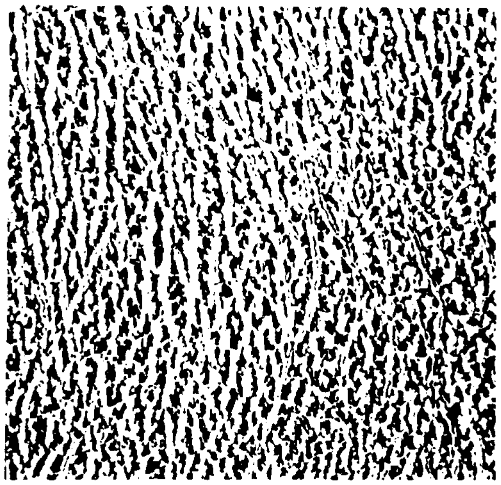
(b) First-order Linear Model



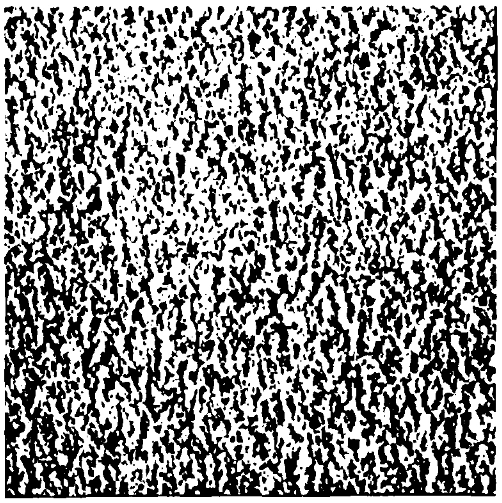
(c) Second-order Linear Model

(d) Second-order Linear Model with Non-stationary Noise

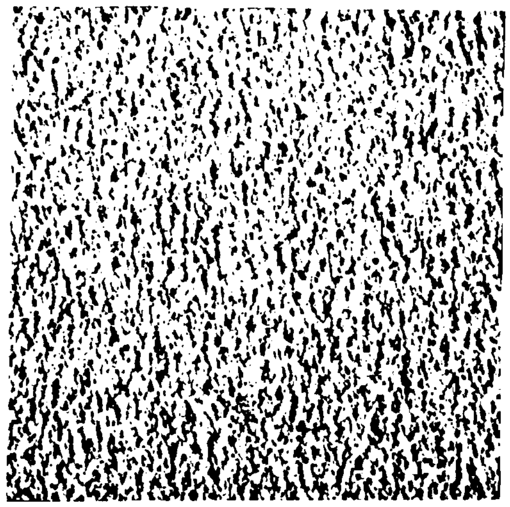
Figure 5.5 Wool



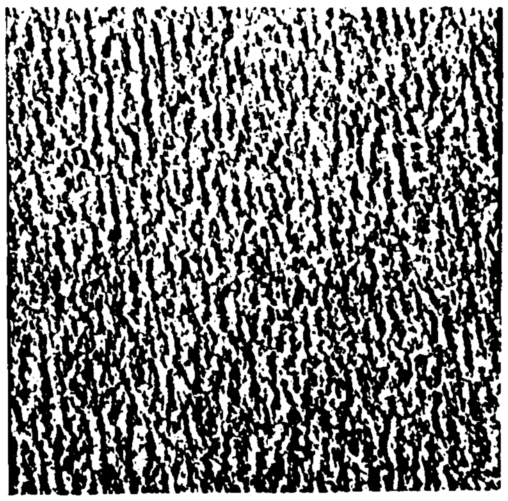
(a) Original Texture



(b) First-order Linear Model

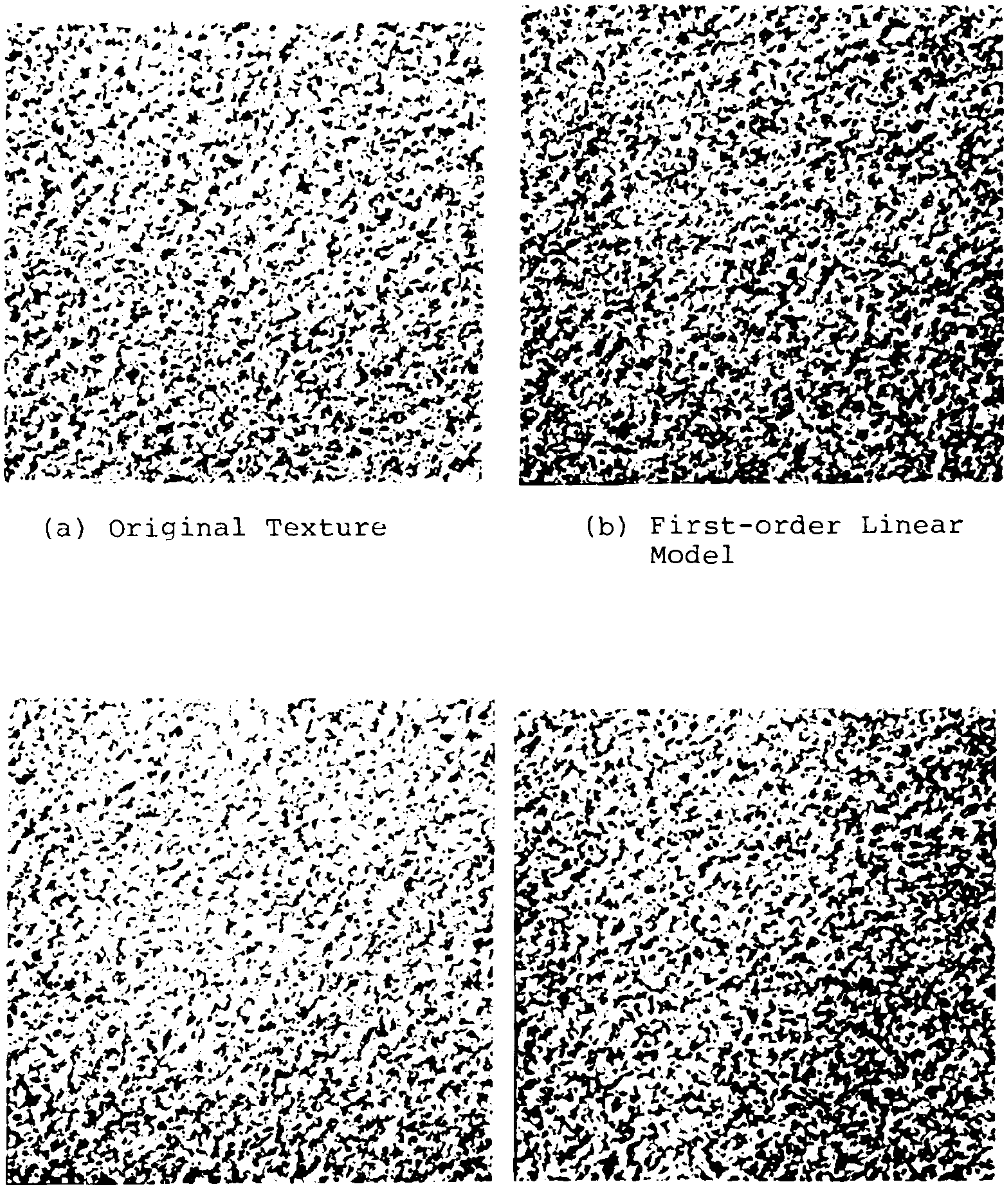


(c) Second-order Linear Model



(d) Second-order Linear Model with Non-stationary Noise

Figure 5.6 Leather



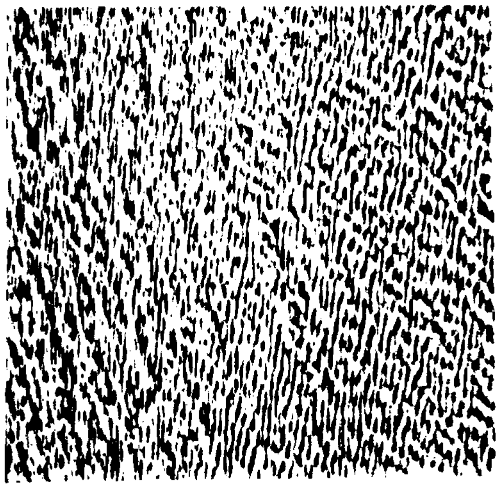
(a) Original Texture

(b) First-order Linear Model

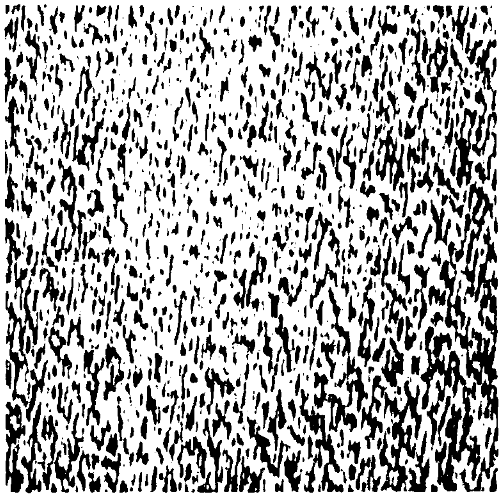
(c) Second-order Linear Model

(d) Second-order Linear Model with Non-stationary Noise

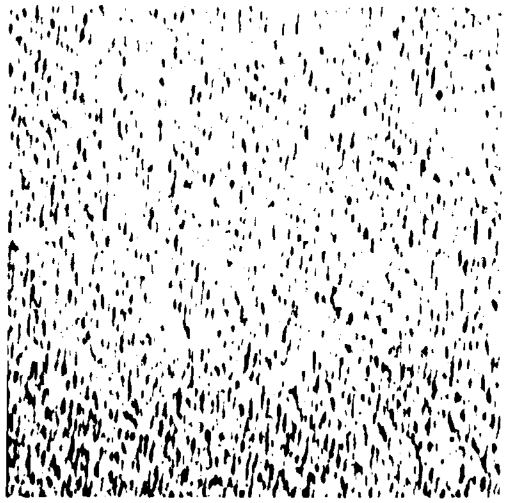
Figure 5.7 Sand



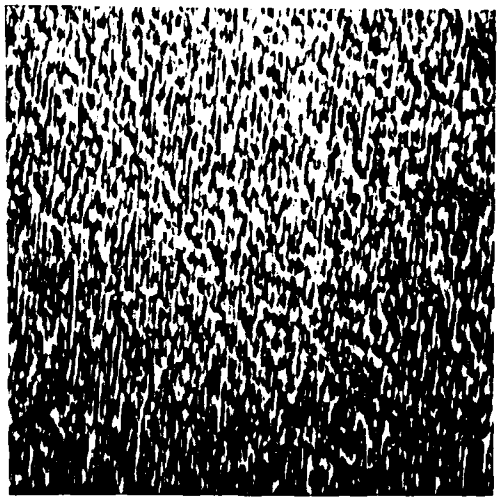
(a) Original Texture



(b) First-order Linear Model

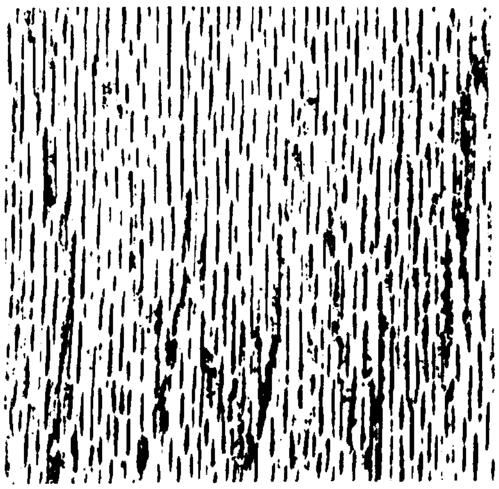


(c) Second-order Linear Model

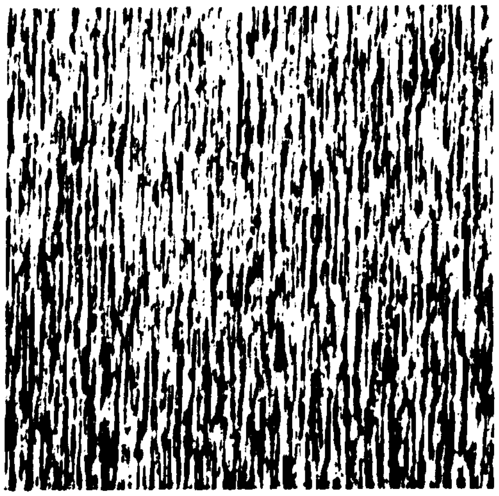


(d) Second-order Linear Model with Non-stationary Noise

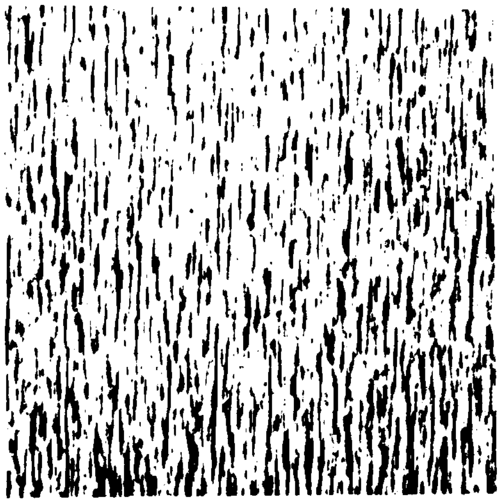
Figure 5.8 Water



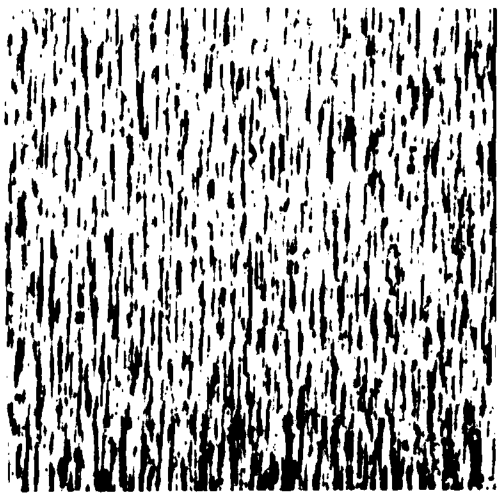
(a) Original Texture



(b) First-order Linear Model

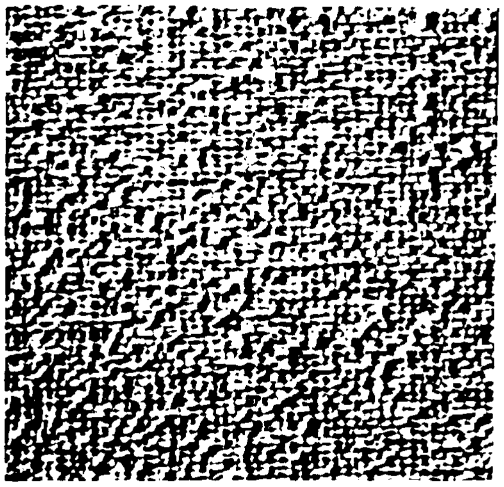


(c) Second-order Linear Model

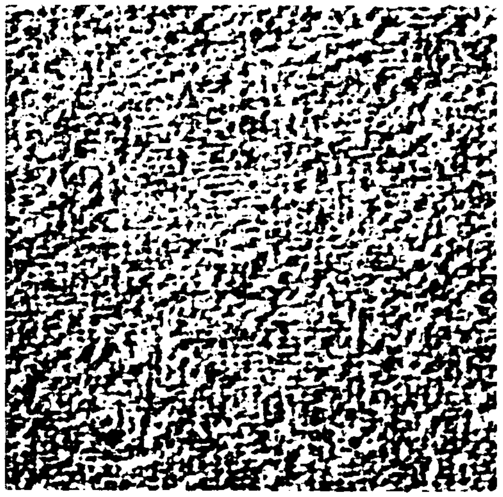


(d) Second-order Linear Model with Non-stationary Noise

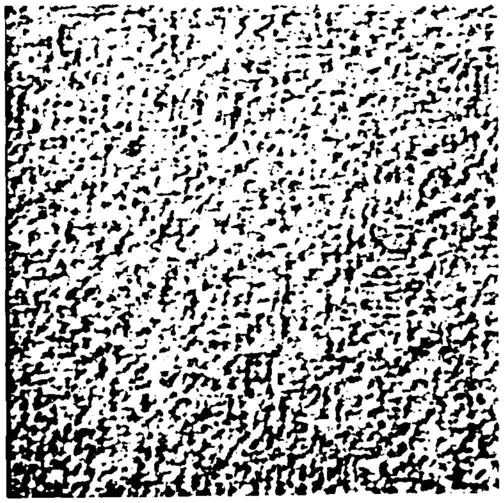
Figure 5.9 Wood



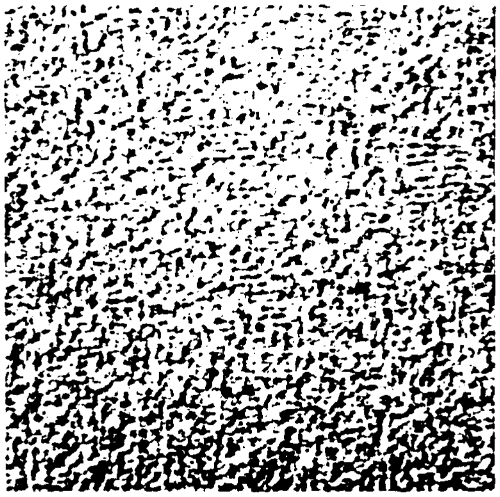
(a) Original Texture



(b) First-order Linear Model

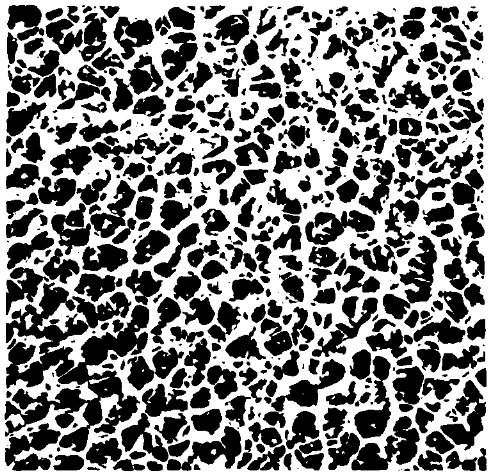


(c) Second-order Linear Model

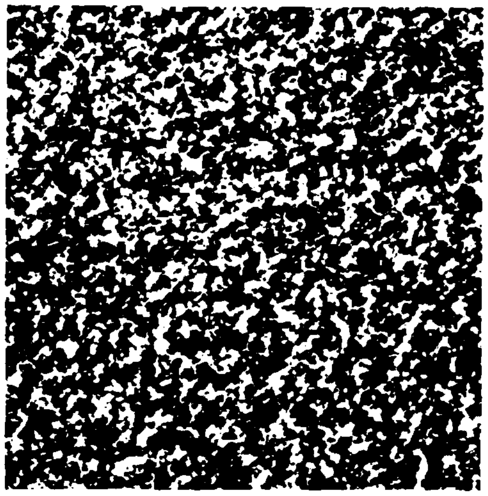


(d) Second-order Linear Model with Non-stationary Noise

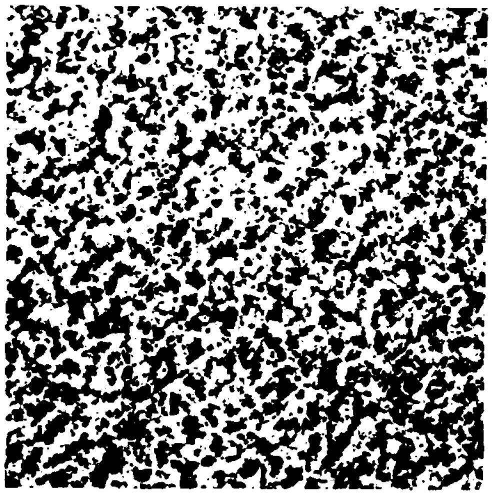
Figure 5.10 Raffia



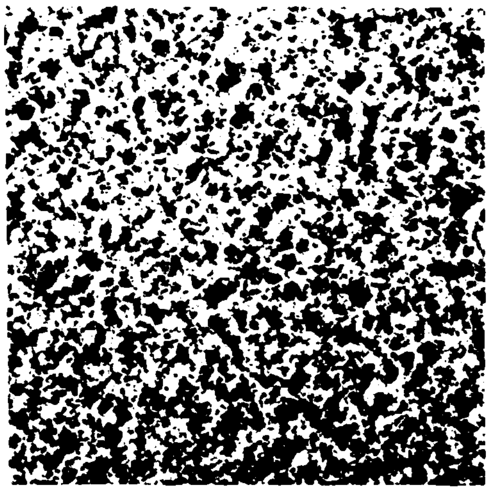
(a) Original Texture



(b) First-order Linear Model



(c) Second-order Linear Model



(d) Second-order Linear Model with Non-stationary Noise

Figure 5.11 Bubbles

simulation of sand when the texture is examined in detail on a high-resolution display device.

5.6 Rotation and Magnification

The effects of image texture rotation and scale changes on the covariance function of a texture can be determined by expressing each as a linear transformation of the linear model. Using the notation of the linear model as expressed by Eq. (5.1) we may define

$$\bar{\vec{X}} = \frac{1}{M}\sum_{k=1}^{M}\vec{X}_k \quad . \qquad (5.32)$$

Using this notation, the maximum likelihood estimators for the mean and covariance matrix of N-variate normal distribution, $N(\vec{X}:\vec{\mu},\Gamma)$

$$N(\vec{X}:\vec{\mu},\Gamma) = (2\pi)^{-N/2}|\Gamma|^{-1/2}\exp[-1/2(\vec{X}-\vec{\mu})^T\Gamma^{-1}(\vec{X}-\vec{\mu})] \qquad (5.33)$$

are

$$\hat{\vec{\mu}}_X = \bar{\vec{X}} \qquad (5.34)$$

and

$$\hat{\Gamma}_{\vec{X}} = \frac{1}{M}\sum_{k=1}^{M}(\vec{X}_k-\bar{\vec{X}}_k)(\vec{X}-\bar{\vec{X}})^T \quad . \qquad (5.35)$$

Relating this to the facts discussed in sections 3.4, 3.5

and $\vec{\omega}$, we see that the linear model parameter estimates are obtained from the maximum likelihood estimators for the mean and covariance matrix.

If we then define a linear transformation H such that

$$\vec{\omega}_k = H\vec{X}_k \tag{5.36}$$

Then

$$\hat{\vec{\mu}}_{\vec{\omega}} = \frac{1}{M}\sum_{k=1}^{M} H\vec{X}_k = H\hat{\mu}_{\vec{X}} \tag{5.37}$$

and

$$\begin{aligned}\hat{\Gamma}_{\vec{\omega}} &= \frac{1}{M}\sum_{k=1}^{M}(H\vec{X}_k - H\bar{\vec{X}}_k)(H\vec{X}_k - H\bar{\vec{X}})^T \\ &= \frac{1}{M} H \sum_{k=1}^{M}(\vec{X}_k - \bar{\vec{X}}_k)(\vec{X}_k^T H^T - \bar{\vec{X}}^T H^T) \\ &= H \frac{1}{M}\sum_{k=1}^{M}(\vec{X}_k - \bar{\vec{X}}_k)(\vec{X}_k - \bar{\vec{X}})^T H^T \\ &= H\hat{\Gamma}_{\vec{X}} H^T \quad .\end{aligned} \tag{5.38}$$

Thus if our model is transformed linearly, the estimates required for linear model parameter estimation can be obtained from original model estimates and the transformation itself.

Rotation may be thought of as a linear transformation of coordinate systems. Where our discrete image is considered to be rotated about an axis by angle θ and the standard row, column notation is used ($I(n_1,n_2)$ represents the pixel value for image I at row n_1 and column n_2)

$$I_{rotated}(n_1,n_2) = I_{original}(n_2\sin\theta+n_1\cos\theta, n_1\sin\theta+n_2\cos\theta) \quad . \tag{5.39}$$

Usually the row, column addresses in the original image are fractional. Therefore, these pixel values must be specially defined. A widely accepted practice is to estimate each value as a function of pixels surrounding it. The most likely candidates are nearest-neighbor and bilinear interpolation[42]. It will be shown that in either case, the new value may be expressed as a linear combination of values in the original image.

A very similar result may be derived in scale changes of a texture. Here

$$I_{scaled}(n_1,n_2) = I_{original}(n_1 \cdot a, n_2 \cdot a) \quad . \tag{5.40}$$

The origin of the coordinate system defines the center of the image magnification or reduction. In rotation, the origin of the coordinate system defines the axis of rotation. In both image magnification and rotation, the row and column addresses of the pixel in the rotated image may be fractional. Again, the value may be expressed as a linear combination of values in the original image.

Consider the example shown in Fig. 5.12. With angle of rotation θ, origin V_5 and magnification of 4/3, using bilinear interpolation (between rows the columns) we have

$$\begin{aligned}
W_1 &= [0.65\ V_1+0.35\ V_4]\cdot 0.375+[0.65\ V_2+0.35\ V_5]\cdot 0.625\\
W_2 &= [0.375\ V_2+0.625\ V_5]\cdot 0.35+[0.375\ V_3+0.625\ V_6]\cdot 0.65\\
W_3 &= V_5\\
W_4 &= [0.625\ V_4+0.375\ V_7]\cdot 0.65+[0.625\ V_5+0.375\ V_8]\cdot 0.35\\
W_5 &= [0.35\ V_5+0.65\ V_8]\cdot 0.625+[0.35\ V_6+0.65\ V_9]\cdot 0.375 \quad .
\end{aligned} \tag{5.41}$$

This implies that

$$H = \begin{bmatrix}
0.24375 & 0.40625 & 0.0 & 0.13125 & 0.21875 & 0.0 & 0.0 & 0.0 & 0.0\\
0.0 & 0.13125 & 0.24375 & 0.0 & 0.21875 & 0.40625 & 0.0 & 0.0 & 0.0\\
0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 0.0\\
0.0 & 0.0 & 0.0 & 0.40625 & 0.21875 & 0.0 & 0.24375 & 0.13125 & 0.0\\
0.0 & 0.0 & 0.0 & 0.0 & 0.21875 & 0.13125 & 0.0 & 0.40625 & 0.24375
\end{bmatrix} \tag{5.42}$$

The accuracy of the estimate depends on the ability of the interpolating function to accurately estimate the value of off-grid samples in a discrete image. The covariance function itself is being interpolated in this method. Caution should be exercised when using a nearest-neighbor approach as the transformation of a non-singular covariance matrix can be singular if the rows of H are not independent vectors.

By rotating covariance matrices, we are able to produce models which can generate textures at any angle and magnification from one matrix. This could also be useful when trying to identify textures of different orientation and scale based on covariance statistics.

5.7 Second-Order Linear Model

When we say that a model is a linear or nonlinear, we are referring to linearity or nonlinearity in the parameters. The value of the highest power of an independent variable in the model is called the order of the model. For example,

$$Y = \beta_1 V_1 + \beta_{11} B_1^2 + \beta_0 + \varepsilon \tag{5.43}$$

is a second-order linear model. A general second-order linear model with two independent variables may be written as

$$Y = \beta_1 V_1 + \beta_2 V_2 + \beta_{11} V_1^2 + \beta_{22} V_2^2 + \beta_{12} V_1 V_2 + \beta_0 + \varepsilon . \tag{5.44}$$

A full second-order model with N independent variables will employ $(N^2+3N)/2$ terms in addition to the β_0 (constant) and ε (error) terms. This general second-order linear model may be written as

$$\begin{aligned} V_{N+1} &= \beta_1 V_1 + \beta_2 V_2 + \ldots + \beta_N V_N + \beta_0 + \beta_{11} V_1^2 + \beta_{12} V_1 V_2 + \ldots + \beta_{NN} V_N^2 + \varepsilon \\ &= \sum_{i=1}^{N} \beta_i V_i + \sum_{i=1}^{N} \sum_{j=i}^{N} \beta_{ij} V_i V_j + \beta_0 + \varepsilon . \end{aligned} \tag{5.45}$$

Second-order models have been particularly useful in studies where surfaces must be approximated by polynomials of low order. In all cases, a second-order model will "fit" given data as well as or better than a first-order model that is a subset of second-order models. This does not imply that the second-order model will be more correct however, as the process which we are attempting to model may be in fact a linear first-order process or some other type.

The use of a second-order model to approximate the surface of the general stochastic model could have many advantages over a first-order model. An example of fitting such a model in one dimension to a given set of data is shown in Fig. 5.13.

Still the linear first-order model may provide a good fit to the data and the magnitude of the unexplained variance in the data may be large enough that the improvement due to the addition of second-order terms to the model may be barely noticeable. In two dimensions, the fitting problem is one utilizing a quadric surface such as a elliptic paraboloid or hyperbolic paraboloid versus a plane to fit a given set of data. Again, the fit

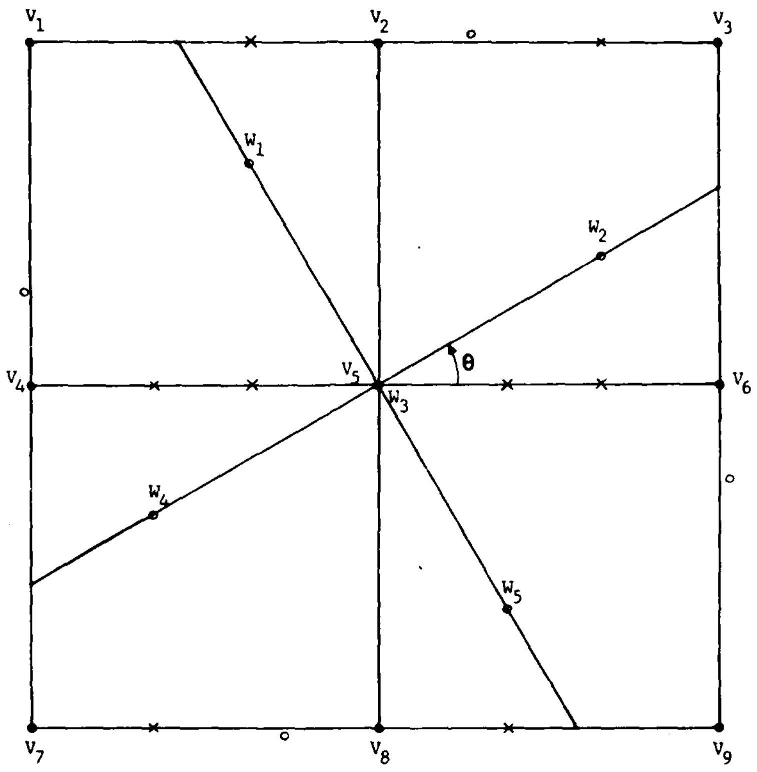


Figure 5.12 Image Rotation and Magnification

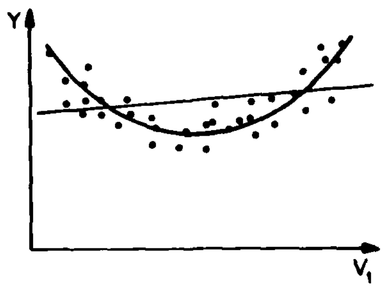


Figure 5.13 Second-order Linear Model

may or may not be markedly better. Adding second-order terms to a model will always produce a fit as good as or better better than a first-order model but the number of computations required to compute the coefficients and fit the model are much greater.

It is also important to note that the covariances of the V_i are required in order to obtain least-square estimates of the parameters β_i in the first-order model [20]. Covariance is essentially a second-order statistic. Therefore, estimating the parameters of a second-order model will require the use of fourth-order statistics. Specifically the correlation of terms $V_{i_1} V_{i_2}$ and $V_{i_3} V_{i_4}$ is needed. This may cause serious problems as many cases the variables in a second-order model will be highly intercorrelated. For example, the terms V_1, V_1^2 and $V_1 V_i$ (if V_i is highly related to V_1) may be strongly correlated. This situation, often referred to as multi-collinearity, may cause problems during the inversion or decomposition of the estimated correlation matrix, a necessary step in model parameter estimation. For this reason, care should be exercised during the analysis of second-order models.

Inside a circular radius of 14 pixels from V_{N+1} there are 307 pixels. To search all possible cross products in this region to find the most significant would require over 47,000 cross products to be examined. Computation of a covariance matrix containing all of these terms is impossible (in practice). In our study we were limited to investigate only 820 possible cross products for entry into the generation model. As most of the variance was explained by the linear terms of the model, most of the cross products were insignificant from a statistical point of view. This selection procedure is detailed in [20] and in Chapter 3. Those that were significant were entered into the model and a new texture was generated using Eq. (5.45) with stationary Gaussian noise and having zero mean and fixed variance [56].

The results of texture simulations using the second-order linear model are shown in Figs. 5.1(c) to 5.11(c). On some of these textures only a slight improvement from the addition of second-order terms may be seen. In most cases, no change can be observed even when the results are displayed on a high-resolution display device. The lack of improvement could be due to the small number of cross-terms examined; however we feel that this number is sufficiently large to show any considerable improvement due to the addition of second-order terms to the linear model.

5.8 Textures with Non-Stationary Noise

Applying a texture generation model to the original parent texture image data used to estimate its parameters gives a residual error image. When applying the linear model to a two-dimensional texture, a two-dimensional image containing the pixel differences or residuals $\hat{V}_{N+1}-V_{N+1}$ is found. Here $\hat{V}_{N+1}$ is the prediction of the next pixel in the sequence as a linear function of the pixel around it according to the model without any noise added. Naturally, we would expect these errors to be small as merely subtracting one pixel from its nearest-neighbor would yield a small value in most natural, low-noise images. Such an image of residuals was generated for the sand and linearly rescaled to show the detail present in the image (Fig. 5.14). Definite patterns are seen to exist in this image and thus a violation of the independent assumption is indicated. Ideally, this residual image would be uncorrelated noise.

A histogram showing the number of $\hat{V}_{N+1}$ occurring at each pixel value is shown in Fig. 5.15. A plot indicating the mean of the residuals $\hat{V}_{N+1}-V_{N+1}$ versus V_{N+1} is shown in Fig. 5.16. As would be expected, residuals where the $\hat{V}_{N+1}$ is less than 0 will have a mean less than zero and those residuals where the V_{N+1} is greater than 255 will be

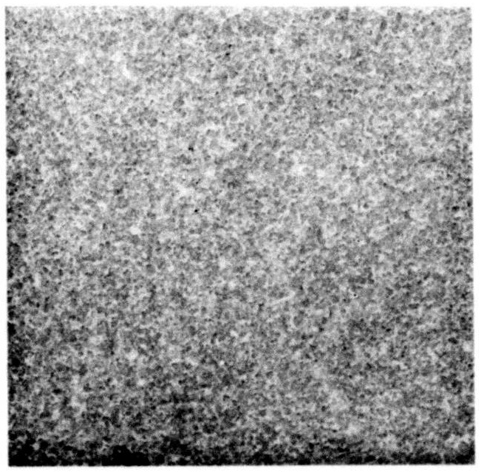


Figure 5.14 Residual Sand $(\hat{V}_{N+1}-V_{N+1})$

likewise positive. Figure 5.17 shows a similar plot of the standard deviation of the residuals versus $\hat{V}_{N+1}$. These three figures seem to indicate that the distribution of the error in the model is related to the value $\hat{V}_{N+1}$. Therefore the assumption of constant error variance is questionable. It may be reasonable to drive the generation process with noise which does not have stationary mean or variance. The effect of such a change in the generation process was studied. Figure 5.18 shows the distribution of error $\hat{V}_{N+1}-V_{N+1}$ (a histogram of the residual image) which appears to be aproximately normal.

The distribution of this error and the relationships between the predicted and actual pixel values was utilized to generate textures using non-stationary noise. The procedure begins by generating a pixel $\hat{V}_{N+1}$ according to Eq. (5.45) excluding the error term. With this predicted value a random error value ε is chosen to be added to $\hat{V}_{N+1}$. This error value ε is chosen from the distribution of error as a function of $\hat{V}_{N+1}$ and can have any arbitrary distribution. The next pixel will than be computed in a similar manner. Results of texture synthesis formed using this model are shown in Fig. 5.1(d) through Fig. 5.11(d).

The arbitrary distribution of error as a function of $\hat{V}_{N+1}$ is calculated by applying the calculated linear model

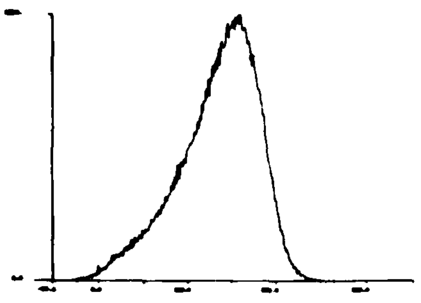


Figure 5.15 Histogram of $\hat{V}_{N+1}$

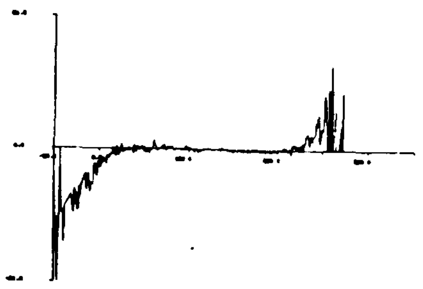


Figure 5.16 Mean of $V_{N+1}-\hat{V}_{N+1}$ vs $\hat{V}_{N+1}$

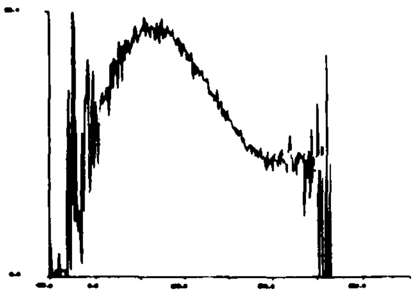


Figure 5.17 Standard deviation of $V_{N+1}-V_{N+1}$ vs. $\hat{V}_{N+1}$

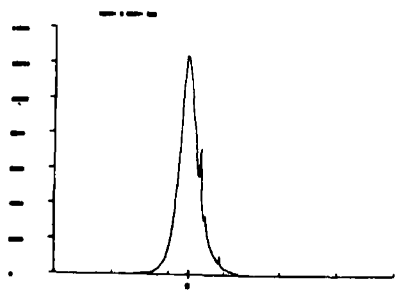


Figure 5.18 Histogram of residual image.

to the original parent texture and computing a histogram of errors as a function of $\hat{V}_{N+1}$.

In most cases, considerable improvement is seen when these simulations are critically observed on a high-resolution display device and compared with the stationary model results. Of course, the information required to generate them is considerably greater also. The distribution of errors as a function of $\hat{V}_{N+1}$ must be condensed and coded to some degree to minimize storage requirements. For a 256-grey-level image $\hat{V}_{N+1}$ usually ranges from -50 to 305 and the errors, $\hat{V}_{N+1}-V_{N+1}$, from -255 to +255. These ranges were determined experimentally. This would yield quite a large amount of data if fully stored. By storing a small number (under 100), typical errors for each range (and not each single value) of $\hat{V}_{N+1}$ the number of data values we are required to store can possibly be reduced to under 1000. Therefore it is believed that this approach of using non-stationary, non-Gaussian noise to generate textures may be quite acceptable even with severe storage limitations.

5.9 Conclusion

The results in this chapter indicate that many natural textures are well simulated using a large autoregressive model. Adding second-order terms to the model improves the results slightly but the resulting increase in computational complexity makes this second-order simulation method difficult to implement. Using non-stationary noise in the generation process improves the simulations considerably when the textures are viewed critically. The subsequent increase in storage and computation required by this addition is small and therefore this model modification should be considered in most applications.

CHAPTER 6

MULTIPLE MODEL TEXTURE GENERATION

6.1 Introduction

In this chapter, we will present three methods of generating textures using multiple texture models. The first method introduces a set of generation kernels that is used to synthesize texture pixels in a multiple-pass manner. Associated with each of these kernels is a unique model. This method could be useful in generating textures which have very coarse structure. The second method uses a piecewise-linear method of fitting the model to parent texture data. The model chosen during the generation process is allowed to depend on the pixel values in the kernel. Although the fit of the model is better, the synthesis results show little improvement over the single-model approach with a linear model. The third method of texture generation presented in this chapter uses a additional image to determine the model number to be used during the generation process. This composite generation method could be useful when a texture is actually composed of a set of subtextures as it allows a unique model for each of these subtextures to be used in the generation process.

6.2 Skip-Generate Method

Simulating textures which have a fine structure is usually a much easier process than simulating textures with coarse structure. This occurs because the linear model contains fewer terms if the texture pixels become uncorrelated over a small distance. For the same texture at a greater magnification, the pixels become highly correlated and the linear model will be forced to contain more terms. As the texture becomes more coarse, more time-consuming statistical measurements must be taken on the parent texture over larger windows. Motivated by these problems, the texture generation algorithms in this section have been developed.

In the texture work so far, pixel V_{N+1} was generated based on pixels above or to the left of it (see Fig. 3.1(b)). As discussed in Chapter 3, the kernel does not have to be contiguous. This kernel shape is chosen to insure that the image space of our synthesized texture was filled during the generation process. However, generating pixels along a row, row by row is not the only way of filling an image space.

Consider the non-contiguous kernel mask in Fig. 6.1. If the spacing between the pixels in this mask is 8, using the linear model in Eq. (5.4) to generate the right-most pixel in the bottom row, we can generate every 8th pixel along every 8th row. At each step the next pixel is generated based on the previously-generated pixels around it (ignoring boundary conditions). After generating an image with this type of spacing, the pixels midway between the previously-generated pixels on each row may be generated using the mask in Fig. 6.2. In this mask, the pixel with the "x" in it denotes the next pixel, V_{N+1}, to be generated according to Eq. (5.4). Naturally, the linear model used in this step will have different coefficients than the previous one. It is also interesting to note that new pixels depend not only on previously generated pixels above them (as with the mask in Fig. 3.1(b)) but depend also on the pixels below them. Still, ignoring boundary conditions, each pixel depends only on previously generated pixels. At the next step a mask similar to that in Fig. 6.3 can be used to fill in the pixels midway between the previously-generated pixels in each column. Again pixels are allowed to depend on pixels around them.

By repeatedly using the masks in Fig. 6.2 and Fig. 6.3 with successively closer and closer pixel

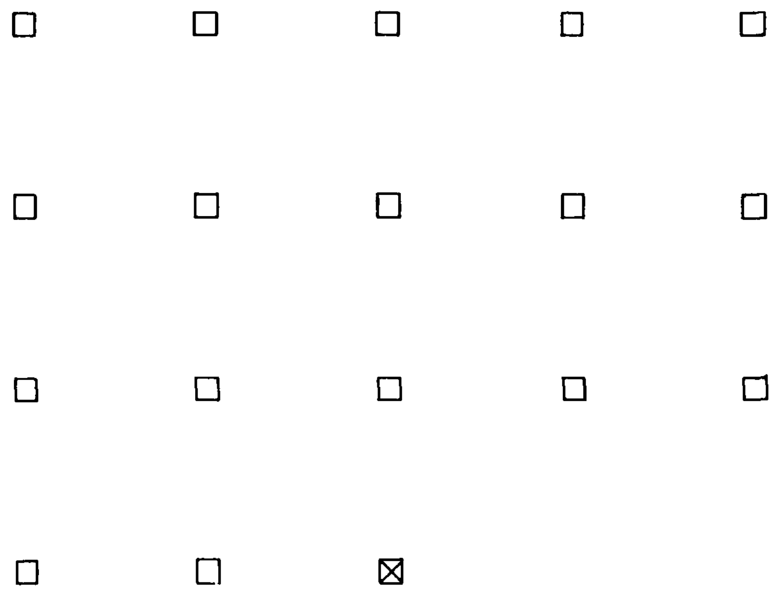


Figure 6.1 First-pass Generation Kernel

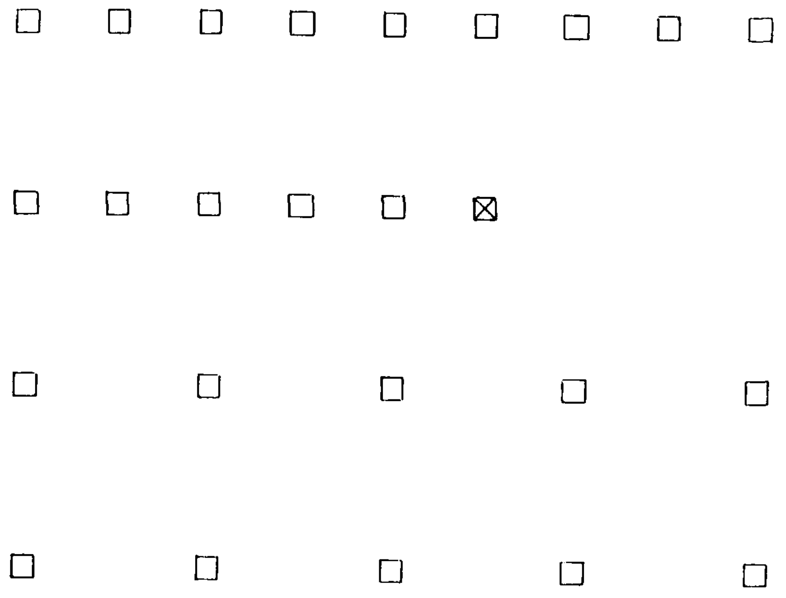


Figure 6.2 Second-pass Generation Kernel

spacing, the texture simulation image space is filled. An example showing the pixels generated at each successive pass is shown in Fig. 6.4. More importantly, to determine the linear model for each mask, only one covariance matrix is required and can contain as many or as few terms as desired. The process of collecting statistics for one matrix is not beyond the complexity that we would want to undertake for the small number of times required by this process. Naturally, any other stochastic process may be substituted for the linear model. As before, only the measurements required to estimate the parameters corresponding to each mask need to be taken. This number depends on the spacing of the pixels in the first mask, which should be a power of two. Other odd-shaped kernels and kernels whose spacing is not a power of two could be designed to form space-filling sets. Most would require more models to be estimated and would provide little additional information.

Texture simulations using this method are shown in Figs. 6.5-6.12. Only a slight improvement is seen in some of the texture simulations over the synthesis done by the earlier single linear model. Most of these textures are apparently well simulated by a carefully chosen model and the results are not critically dependent on the coarseness of the textures.

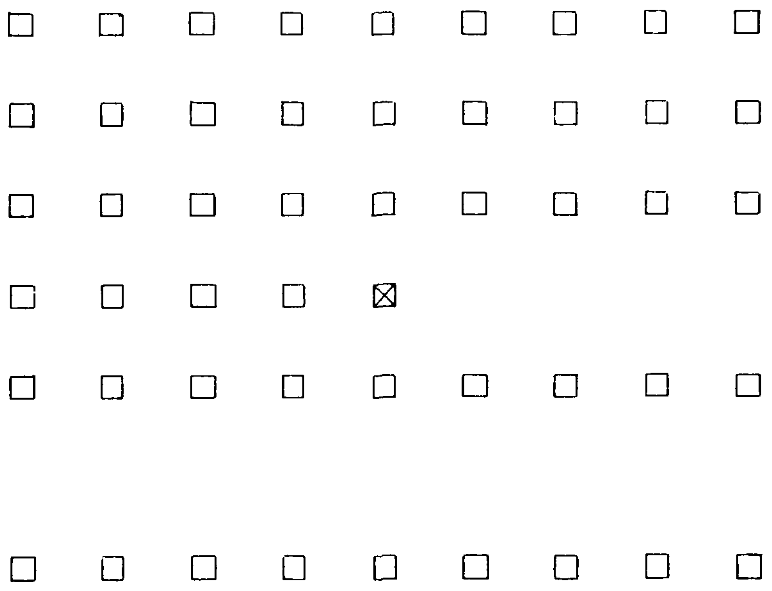


Figure 6.3 Third-pass Generation Kernel

```
1 6 4 6 2 6 4 6 1 6 4 6 2 6 4 6 1
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
3 6 4 6 3 6 4 6 3 6 4 6 3 6 4 6 3
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
1 6 4 6 2 6 4 6 1 6 4 6 2 6 4 6 1
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
3 6 4 6 3 6 4 6 3 6 4 6 3 6 4 6 3
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6 5 6 5 6 5 6 5 6 5
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
1 6 4 6 2 6 4 6 1 6 4 6 2 6 4 6 1
```

Figure 6.4 Filled Space of Skip-generate Method

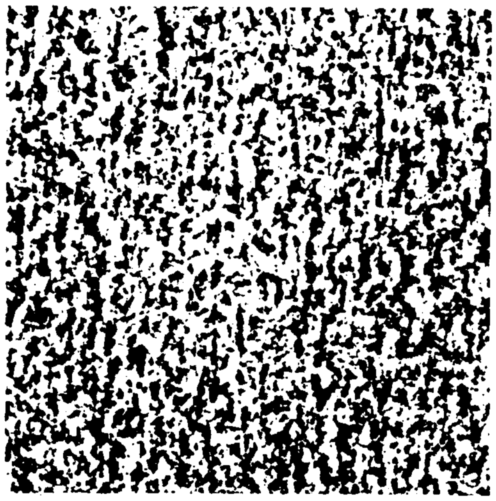


Figure 6.5 Skip-generate Bark

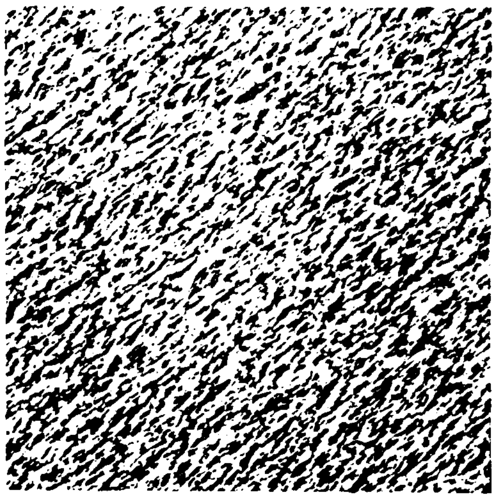


Figure 6.6 Skip-generate Straw

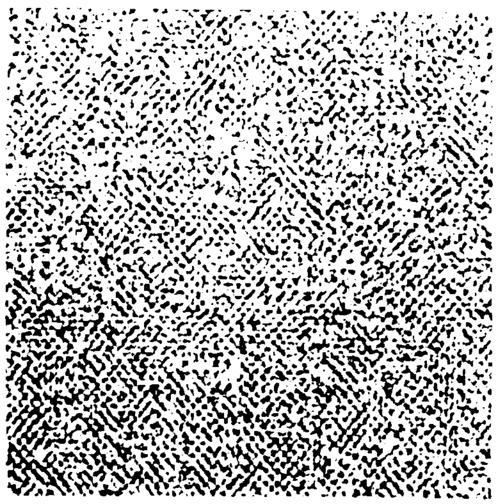


Figure 6.7 Skip-generate Cloth

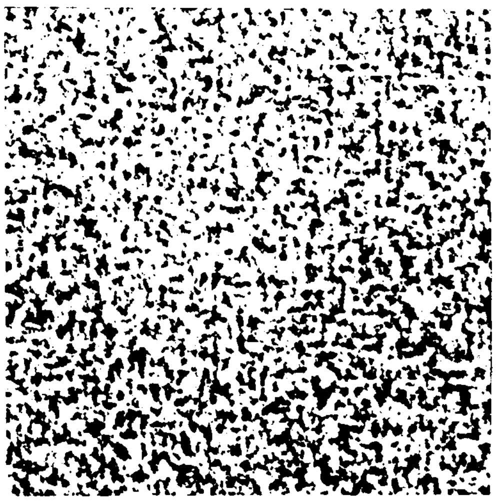


Figure 6.8 Skip-generate Wool

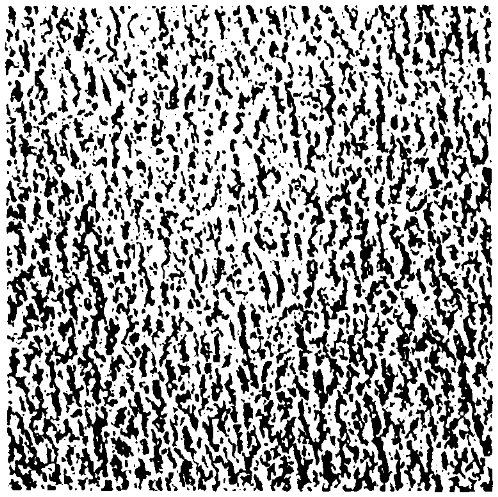


Figure 6.9 Skip-generate Leather

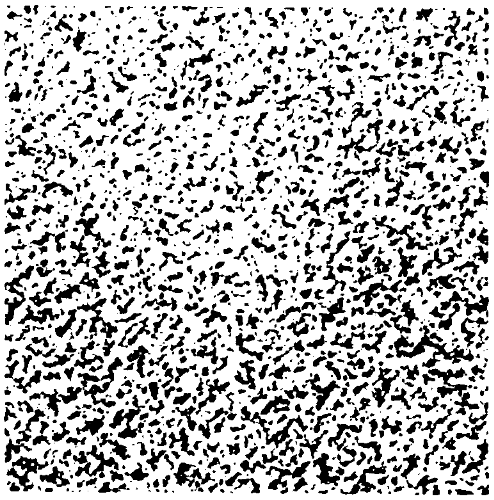


Figure 6.10 Skip-generate Sand

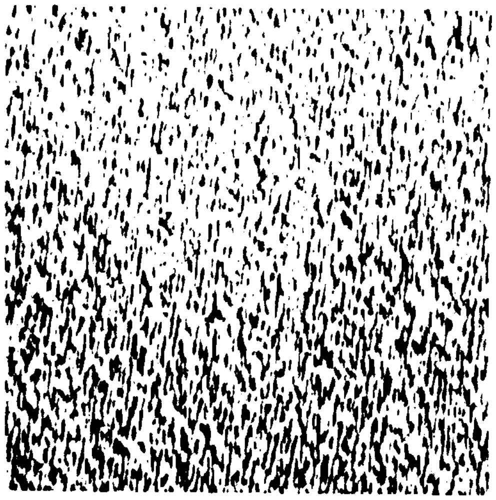


Figure 6.11 Skip-generate Water

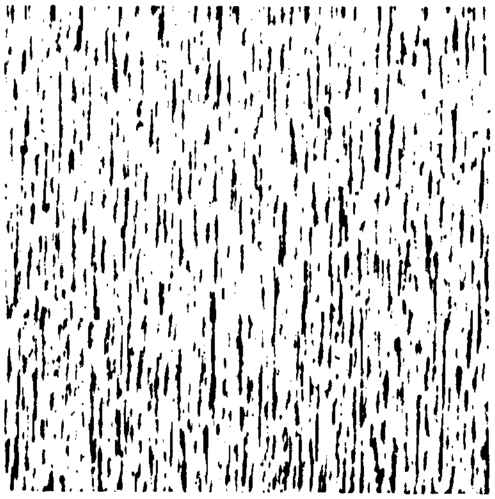


Figure 6.12 Skip-generate Wood

A word of caution should be added concerning the computations involved in the linear model coefficient calculation of this method. During the later stages of the skip-generate method, the pixels in the generation kernel become highly correlated as the distance between them decreases with each pass. This may cause the correlation or covariance matrix of the model to be ill-conditioned. To avoid numerical problems, the number of variables entered into the process, and therefore the number of steps involved in the matrix decomposition process, should be kept to a minimum in some cases. The use of ridge regression techniques [43,44,46] might also be considered.

6.3 Piecewise-Linear Models

When generating textures using the general linear model described by Eq. (5.4) and the generating kernel in Fig. 3.1(b) the same model is used regardless of the values of the pixels $V_1,\dots,V_N$. By developing more than one linear model and allowing the choice of the model at each pixel generation step in the synthesis process to be dependent on some functional value of $V_1,\dots,V_N$, $F(V_1,\dots,V_N)$ a new synthesis model is formed.

To illustrate this concept consider the data in Fig. 6.13(a). If we were to fit one linear model to the data in order to predict V_2 from V_1 it would look like the single line running through the data in Fig. 6.13(a). This linear model could then be used to predict V_2 based on the value of V_1. But if we allow the choice of our linear model to be dependent on the value of V_1, then for an incoming value of V_1 we choose a model whose domain includes V_1 to predict V_2. For 6 linear models, the straight lines are shown in Fig. 6.13(b). The fit to the data using multiple linear models will always be as good as or better than that of the single linear model. That is, the mean square error will generally be reduced using multiple models.

Using multiple linear models for texture synthesis we would generate pixels V_{N+1} based on pixels $V_1, \ldots V_N$ in the following way. First, we compute a function, F, of the $V_1, \ldots, V_N$ pixels which allows us to choose the proper linear model. Then using this model with the values $V_1, \ldots, V_N$ we predict V_{N+1} and add noise. This process is diagramed in Fig. 6.14.

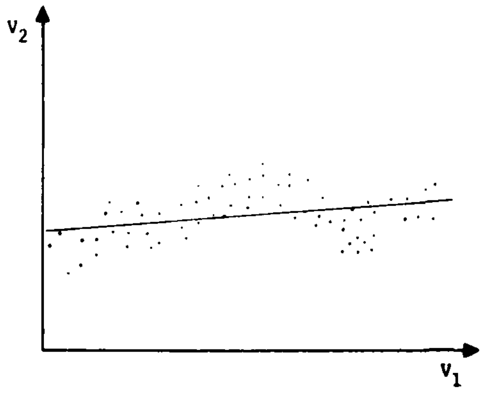
Ideally, the function F should be chosen to minimize the total mean square error resulting from fitting the limited number of models to the sample data. This is very difficult to do in practice however as for N larger than 3 we are fitting multiple hyperplanes to data in an N+1 dimensional space.

One texture synthesis of sand was done using the multiple linear model (see Fig. 6.17). In this case eight models were used and the model number was chosen by examining the pixel immediately to the left of the pixel being generated. The range of this pixel, 0 to 255, was divided into 8 equal subranges and the model was chosen according to the subrange into which the value fell. Only a slight improvement over the single linear model synthesis (see Chapter 5) is seen even though the same kernel shape was used. No other simulations have been run using this model as it is felt that little improvement will result. Also the linear model containing cross-product terms in Chapter 5 probably provides a very good fit in most cases and in more dimensions. In one dimension the model of Chapter 5 would fit the data in Fig. 6.13 with a quadratic curve.

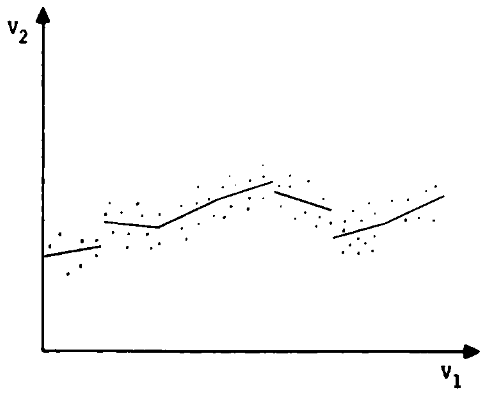
6.4 Field-Definition Stochastic Model

Another method of using multiple stochastic models is to generate an image of fields defining the model number to be used in a second pass. Such an approach would be useful in simulating textures which have multiple sub-textures within them. A simple analytical example is shown in Fig. 6.15. Real world examples might include such things as a brick wall where the texture of the bricks is different than the texture of the concrete separating them. It was felt that this type of approach might be useful in the simulation of bark (see Fig. 3.6(a)) which has a strong macro structure. A method to separate this texture into two fields, which would later define the model to be used, was designed. This result (Fig. 6.18) was obtained by successively passing smart median filters of varying sizes over a binary image (which was obtained by thresholding an original continuous grey-level image) (see Fig. 5.2(a)).

The smart median filters replace the center pixel of a window with the median only if certain conditions are met. The window is passed over the entire image pixel by pixel along a row, row by row in a two-dimensional convolution manner as in Fig. 6.16. Let $I_I(n_1,n_2)$ be the input image and $I_0(n,n)$ be the output. Let the pixel values be 0 (black) and 1 (white). Let N_B denote the number of black pixels in the window being processed with center $I_I(n_1,n_2)$ and let N_W be the number of white pixels in the window. For the binary case, the smart median filter is defined as



(a) Single Linear Model



(b) Piecewise-Linear Model

Figure 6.13 Linear Models

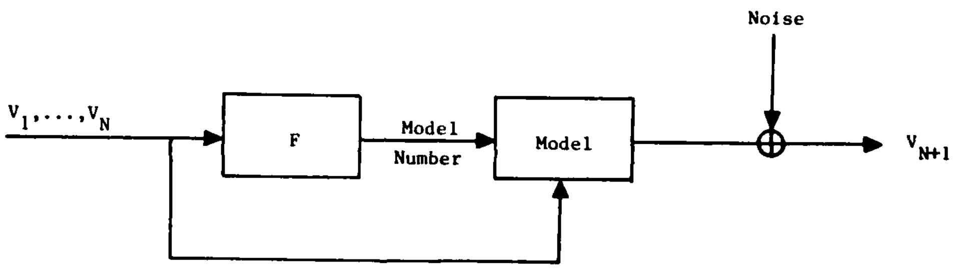


Figure 6.14 Diagram of Multiple Linear Model Method

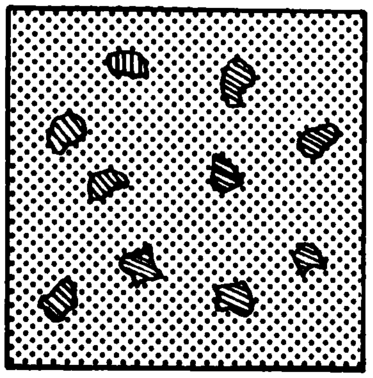


Figure 6.15 Analytic Sub-textures

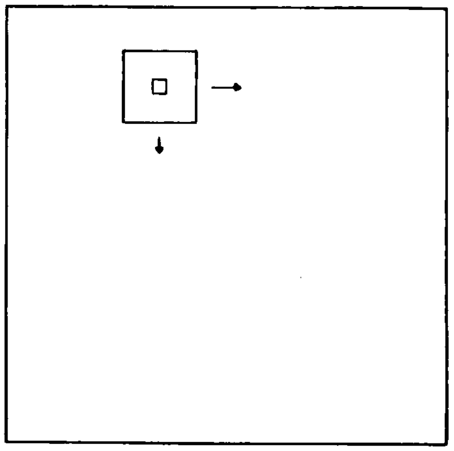


Figure 6.16 Two-dimensional Convolution

$$I_0(n_1,n_2) = \begin{cases} 0, & \text{if } \dfrac{N_B}{N_B+N_W} \geq \text{Thresh} \\ 1, & \text{if } \dfrac{N_W}{N_B+N_W} \geq \text{Thresh} \\ I_I(n_1,n_2), & \text{otherwise} \end{cases} \qquad (6.1)$$

Equation (6.1) indicates that the center pixel of the window is replaced by the median if the percentage of black or white pixels is above a specified threshold. The term "replaced" is used loosely in this context to indicate the visual appearance of replacement when the input and output images are superimposed. In the binary case, the median of the pixels in a window is equal to the value of the most frequently occurring pixel in the window. The threshold and window size varied in the successive passes over the image. The window sizes and thresholds for each of the passes used to obtain Fig. 6.18 were 3-0.50, 3-0.50, 5-0.50, 7-0.75, 11-0.78, 7-0.50. This multiple-pass procedure helped to retain detail and eliminate fields too small for useful measurements.

The field-definition stochastic model was also applied to the non-stationary cloth texture (see Fig. 5.14(a)). This texture is clearly composed of two alternating subtextures. In this case, the texture fields can be extracted by hand (see Fig. 6.21).

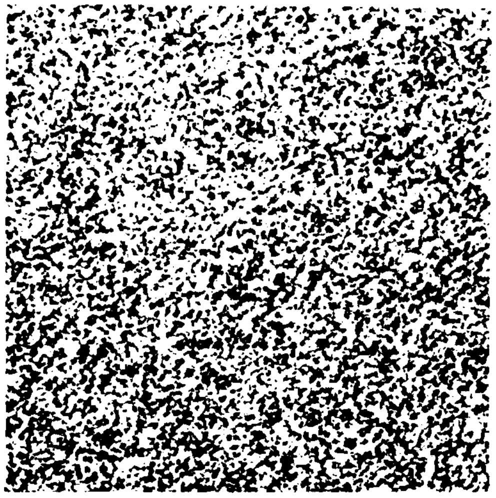


Figure 6.17 Piecewise Linear Model Sand

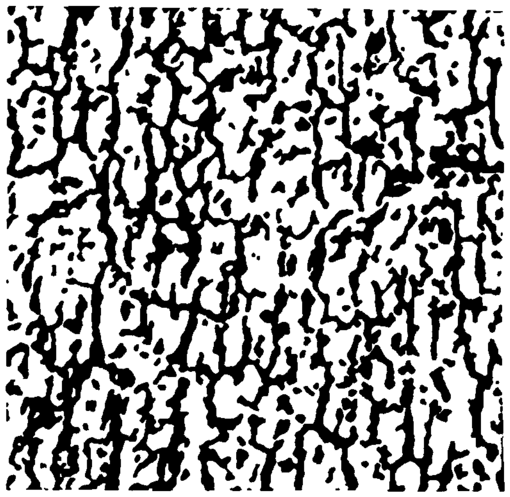


Figure 6.18 Filtered Binary Bark

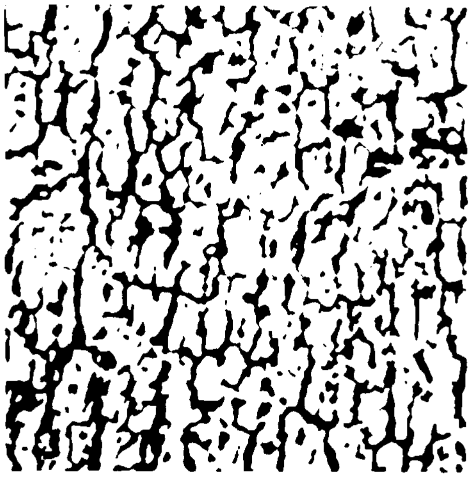


Figure 6.19 Field-definition for Bark

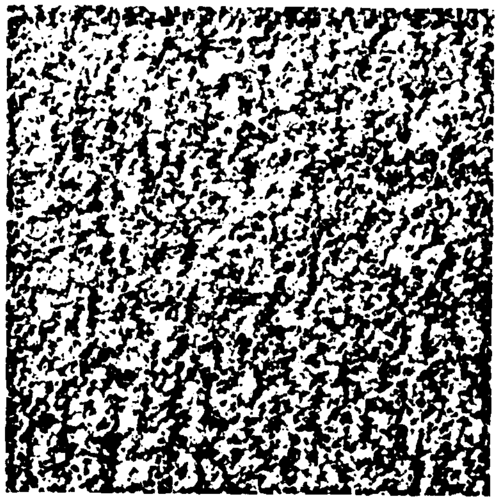


Figure 6.20 Field-definition Bark Simulation

Once the two-field images for bark and cloth were obtained they were processed again to define a total of four fields, the two original fields and two border or transition fields. The transition fields are required as the kernel of points over which relational measurements (usually second-order statistics) are taken may not fall wholly within one of the two fields. Such border measurements will surely increase the inaccuracy of the models for the original fields. So these border regions are considered to be unique texture fields. They are mathematically defined by passing the original linear model texture generation kernel obtained in Chapter 5 over the texture and defining the relative importance of each pixel to be equal to the absolute value of its associated β_i coefficient of the linear model (Eq. (5.4)). These coefficients are usually largest near the eye of the kernel. The "importance" of the eye itself was set to 1.5 $\max(\beta_i)$. If a large percentage (90% for bark, 95% for cloth) of the total "importance" fell within one of the two original fields, the point would be considered as a member of that field. If not, the point would become a member of the transition field based on its position with respect to the two fields, A and B. If, at the point being processed, A is to the left of B then the point is in one transition field, if B is to the left of A then the point is in the other transitional field. The right-left orientation is used as the texture generation process is usually done in a left-to-right manner along each image row.

The results from processing the original field data for the textures bark and cloth are shown in Fig. 6.19 and Fig. 6.22. The four fields are represented using four grey-levels. The black regions on the border may be considered to be undefined as the kernel points do not lie with the image region in these areas.

These field definitions were used both in the statistics gathering process as well as the texture synthesis process using the four calculated models. The models were linear (see Eq. (5.4)) and the methods used in their estimation were discussed earlier in Chapter 5. The synthesis results are shown in Fig. 6.20 and Fig. 6.23. Unfortunately, the field structure is not strongly apparent. The primary reason is that each model requires a number of pixel generations before it reaches a steady state and in most cases this is much greater than the size of most of the fields in the bark texture. By observing the border regions of many synthesis runs, it seems that a steady state is reached after 10-20 pixels have been previously generated. In the cloth texture synthesis of

Fig. 6.23, the randomness of the noise seems to destroy the structure of the texture quite frequently. This is illustrated in Fig. 6.24 where only one of the four models was used to generate the entire texture. Each of the models contained the same set of kernel points used in the original linear model of Chapter 5. It is possible that improvement could be made by choosing a different set of points for each field-model.

In an actual simulation, once a field image is obtained a method must be developed to simulate this field texture and this field texture will then in turn be used to choose the model numbers in the generation of final synthesis. Generating textures with only a few grey levels can be done using more complete stochastic statistics, perhaps N-grams, but the large size of the fields may require that a method such as the skip-generate method (discussed in section 6.2) be used. For many textures, such as cloth, the field generation process would be simple.

More work should be done in the multiple model area as the storage requirements for such models is very small but can potentially produce improved results. A great number of combinations and approaches are possible in this area and, in many cases, may be chosen by the application or the particular textures being simulated.

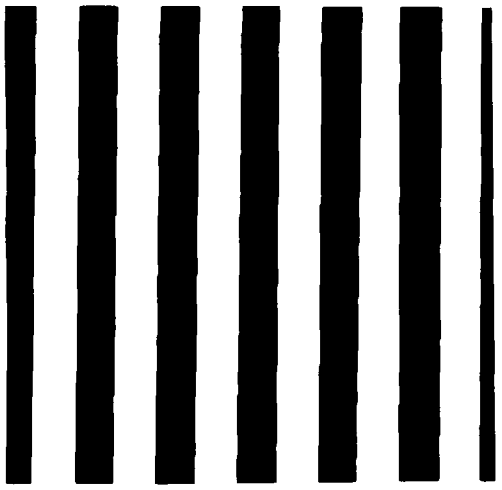


Figure 6.21 Cloth Fields

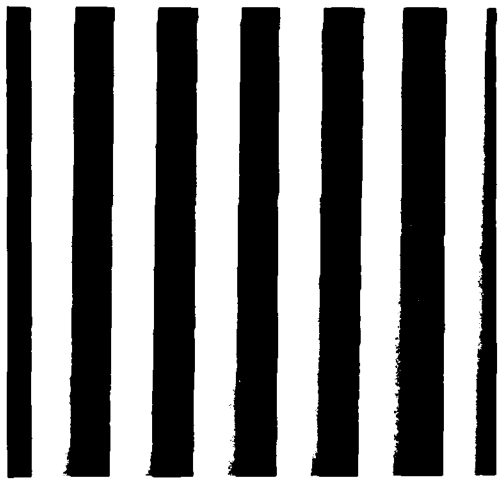


Figure 6.22 Field-definition Image for Cloth Simulation

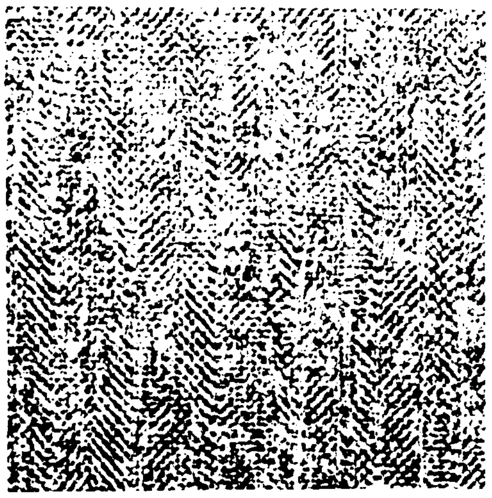


Figure 6.23 Field-definition Cloth Simulation

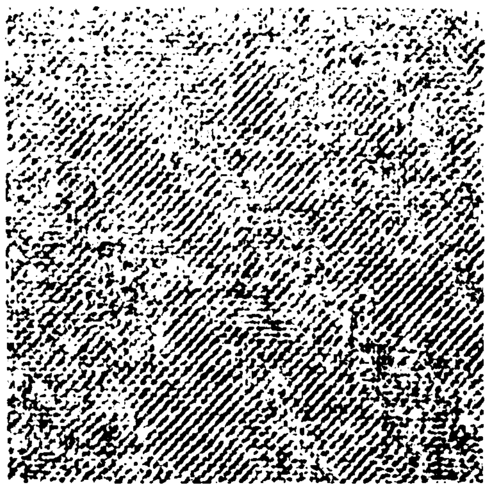


Figure 6.24 Single-field Cloth Texture

CHAPTER 7

BEST-FIT TEXTURE MODEL

7.1 Introduction

A method of generating texture simulations according to their Nth order densities was investigated for binary textures in Chapter 3. The simulations resulting from this Markov process resembled their parental textures quite closely in most cases. When applying a similar concept to multi-grey level imagery, the limits of computer storage are soon reached. To circumvent this constraint, a new method of texture synthesis was developed and applied to a number of textures. Simulation results using this method are given in this chapter.

7.2 N-grams in Continuous Imagery

In binary texture generation based on N-grams a single functional value $P(V_{N+1}/V_1,\ldots,V_N)$ was stored for each possible pattern $(V_1,V_2,\ldots,V_N)$ where the V_i's can be zero or one. This value, also called a generation parameter, represented the conditional probability that the next pixel, V_{N+1}, in the generation process would be a zero-valued, black pixel. The V_i's were chosen by a best linear model fit detailed in Chapter 3 and therefore the kernel of previous pixels $(V_1,...,V_N)$ is not necessarily contiguous (see Figure 7.1). Details concerning the estimation of $P(V_{N+1}/V_1,...,V_N)$ from a parent texture are given in Chapter 3. For binary textures, this single value is sufficient to define the distribution of V_{N+1} given $V_1,...,V_N$. The number of different functions which must be stored is 2^N. In the generation process each pixel V_{N+1} is generated based on the values of the pixels $V_1,...,V_N$ surrounding it and on a computer-generated uniformly-distributed random variable. The texture simulations are generated pixel by pixel along a row until each row is complete. Pixel generation along the edges of an image can be handled in a variety of ways but in Chapter 3 pixels in these border regions were assumed to be any random value, 0 or 1, if they were outside the image boundaries.

A similar approach could be used to generate multi-grey-level textures. For a texture containing g grey levels, g^{N+1} different functions, $P(V_{N+1}/V_1,...,V_N)$, must be stored. (Actually only $(g-1)\,g^N$ are required as $\sum_{x=0}^{g-1} P(X/V_1,...,V_N)=1$ for all V_i). Storage limitations are soon reached. Also estimation of $P(V_{N+1}/V_1,...,V_N)$ is difficult as multiple occurrences of the pixel pattern

$V_1,\ldots,V_N$ may not exist in the parent texture. Therefore even without storage limitations the problems of estimating $P(V_{N+1}/V_1,\ldots,V_N)$ from a given parent texture, which represents the distribution of V_{N+1} given the values of $V_1,\ldots,V_N$ is complex.

This estimation problem no doubt has a number of <u>ad hoc</u> solutions. The problem is basically that for large N and/or large g, there may not be a suitable number of occurrences of the pattern $V_1,\ldots,V_N$ to adequately estimate the distribution $P(V_{N+1}/V_1,\ldots,V_N)$ given a finite sample size. Even though a certain pattern never occurs or rarely occurs in our sample parent texture it is not implied that such a pattern is impossible and will never occur in our simulation synthesis. We might often find numerous occurrences of this pattern if our sample size or the size of our parent texture was increased, especially in noisy and fine-structured textures. But as this very large sample may not be present, we must estimate $P(V_{N+1}/V_1,\ldots,V_N)$ for all $V_1,\ldots,V_N$ based on available samples.

One approach would be to use sample patterns which closely resemble but which may not be exactly the same as each pattern $(V_1,\ldots,V_N)$. That is in a pictorial sense, we use patterns of $(V_1,\ldots,V_N)$ which look "close to" the pattern for which we are attempting to estimate $P(V_{N+1}/V_1,...,V_N)$. Therefore samples in our sample parent texture may be used to estimate numerous $P(V_{N+1}/V_1,...,V_N)$ and not just those they fit exactly. The concept of a distance function must be used to numerically define "close to". Given two patterns, one from our sample texture and the other from the conditional probability of the kernel we are attempting to estimate, the distance measure can be used to determine the value of that sample in estimating $P(V_{N+1}/V_1,...,V_N)$. If the fit between the kernel pattern and the pattern in the sample texture is good the associated value of V_{N+1} in the parent texture will be valuable in estimating $P(V_{N+1}/V_1,...,V_N)$.

Normally, when N and g are small or when we have many samples for any given $V_1,...,V_N$, we can use the histogram of the associated V_{N+1} to estimate $P(V_{N+1}/V_1,...,V_N)$. Here the relative number of times a particular value of V_{N+1} occurs given a pattern indicates the conditional probability we are attempting to estimate. This was discussed in section 3.2. Wnere a distance measure is used instead, a good fit could be considered to be synonymous with high frequency of occurrence of that pattern and a poor fit with low frequency of occurrence.

If such a method of estimating these conditional probabilities is used we are still faced with a huge storage problem. For this method to be practical, the storage requirement must be reduced. From an information standpoint, it is interesting to note that a method of estimating N-grams or conditional probabilities $P(V_{N+1}/V_1,...,V_N)$ from a sample parent texture image produces g^{N+1} data values from M pixel samples where M is the size of the square parent texture image in pixels. For large g and N this is a drastic increase in data. But the actual information content can really never be greater than that content of the sample parent texture image. Therefore, this M value represents an upper bound on the amount of data we should use to generate a texture simulation. Any amount of data exceeding this will contain redundant data.

7.3 The Synthesis Method

Combining this concept of upper bound with the idea of forming a distance measure to compare two texture kernel patterns leads to a new texture synthesis method. In this method, we generate the next pixel based on the pixels in the kernel surrounding it (see Figure 7.1) and their comparison to similar kernels in the parent texture. This comparison is made by passing the kernel currently present in the simulation process over the parent texture and computing the distance function at all possible points (see Figure 7.2). Denoting the pixels in the parent texture by $X_{i,j}$, $i,j=0,\ldots,\sqrt{M}-1$ and the pixels in the kernel $V_1,\ldots,V_N$ by $Y_{i,j}$, we can compute a comparison image

$$C_{a,b} = \text{COMPARISON}(X_{i+a,j+b}, Y_{i,j}) \qquad (7.1)$$

for all a,b such that the kernel is within the boundaries of the texture.

One possible comparison function would be correlation. Assuming, without loss of generality, that our kernel is contiguous as in Figure 7.3 and the elements are denoted as $Y_{i,j}$, this function would be defined as

$$r_{a,b} = \left[\sum_i\sum_j X_{a+i,b+j}\,Y_{i,j} - \frac{\left(\sum_i\sum_j X_{a+i,b+j}\right)\left(\sum_i\sum_j Y_{i,j}\right)}{N}\right] \left[\left[\sum_i\sum_j X^2_{a+i,b+j} - \frac{\left(\sum_i\sum_j X_{a+i,b+j}\right)^2}{N}\right] \left[\sum_i\sum_j Y^2_{i,j} - \frac{\left(\sum_i\sum_j Y_{i,j}\right)^2}{N}\right]\right]^{-1/2} \qquad (7.2)$$

The problem with this particular distance measure is quite serious. Correlation does not take into account differences in over-all mean. For example, the kernels in Figure 7.3 are perfectly correlated but their means differ significantly. Differences in first-order statistics between these kernel patterns will not be detected by a correlation measure and so another comparison function to supplement a correlation value would be required.

A better comparison function would be the mean square difference (MSD). This is defined as

$$MSD_{a,b} = \sum_{i}\sum_{j}(X_{i+a,j+b} - Y_{i,j})^2 \qquad (7.3)$$

where i,j must be within the coordinate range of the kernel as in Eq. (7.1). This comparison function will detect differences in first-order statistics between two pixel patterns (such as those in Figure 7.3) as the MSD function is a sum of squares of differences. Whereas the correlation coefficient of Eq. (7.2) varies between -1.0 and 1.0 and is largest when the fit is good, the MSD measure is small when the fit is good and it is always positive.

The MSD function weights the comparison of all elements in a kernel equally. Having studied many texture generation models we immediately recognize that this fit is not properly weighted. The few pixels which are closest to Y_{NEXT} in proximity are far more important when predicting Y_{NEXT} than those which are far away. So

Eq. (7.2) must be modified to give the weighted mean-square difference (WMSD)

$$\mathrm{WMSD}_{a,b} = \sum_i \sum_j (X_{i+a,j+b} - Y_{i,j})^2 \cdot W_{i,j} \quad . \qquad (7.4)$$

A possible choice for W is

$$W_{i,j} = \frac{1}{(i-i_{NEXT})^2 + (j-j_{NEXT})^2} = \frac{1}{R^2} \qquad (7.5)$$

where R is the euclidean distance between pixel $Y_{i,j}$ and the kernel eye Y_{NEXT} and the coordinates of the eye are given by (i_{NEXT}, j_{NEXT}).

As the first step in comparing a given kernel $Y_{i,j}$ to all kernels in the parent texture, for each point (a,b) in the parent texture, ignoring edges, the WMSD is computed resulting in an image of WMSD's. Where the fit between the generated kernel $Y_{i,j}$ and the image $X_{i,j}$ is good, we would expect $\mathrm{WMSD}_{a,b}$ to be small. The smallest WMSD represents the "best" fit according to our norm. We could choose the Y_{NEXT} associated with this best fit at point (a,b) to be our next pixel in the generation process, however this can cause problems. First of all, the generation process would "lock in" on the parent texture and the generated texture could very well become just an exact copy of the input parent texture. Second, we know ideally that Y_{NEXT} has a distribution, not just a mean. In the autoregressive model of Chapter 5 we gave Y_{NEXT} a distribution by adding random noise to it. Although this could be done here, such an approach would fail to use additional information contained in the WMSD image. There may be a set of points (a,b), all exhibiting a good fit to the kernel pattern $Y_{i,j}$. In fact, the best fit may have a noisy Y_{NEXT} and the other good fits could provide information to improve the prediction of the Y_{NEXT} in the generation process. Using a set of best fits is equivalent to increasing our sample size. We look at a set of similar patterns to pick our Y_{NEXT}.

At this point there are numerous ways to proceed. Logically those patterns with the "best" fit should provide better estimators for Y_{NEXT} so some kind of weighting decision is needed to choose the relative importance of the WMSD's found. If we search through the WMSD image and find the minimum value, $WMSD_{min}$, and scale all the WMSD's by that we form a new image MAX1

$$MAX1_{a,b} = \frac{WMSD_{min}}{WMSD_{a,b}} \quad . \tag{7.6}$$

This image has the value 1.0 at the best fit point and values $0 \leq MAX1 \leq 1.0$ elsewhere.

Here we can look at the MAX1(a,b) image and study its range. If $0.16 \leq \text{MAX1} \leq 1.0$ it is implied that the worst fit yields a 0.16 value. Somehow that worst fit should be translated to imply that the probability of choosing the Y_{NEXT} associated with that point $(a,b)_{WORST}$ is nearly 0.0. The simplest way of doing that is to take powers of the image MAX1(a,b). The maximum remains 1.0 while smaller numbers approach 0.0. For example $(1.0)^{10}=1.0$ but $(0.16)^{10} = 1\text{x}10^{-8}$. We do this to obtain an <u>ad hoc</u> estimate of $P(Y_{NEXT}/Y_{i,j})$. After experimentally studying the values of MAX1(a,b) and its powers, the value of 16 was chosen and a new image PDFUNS

$$\text{PDFUNS}_{a,b} = (\text{MAX1}_{a,b})^{16} \tag{7.7}$$

was used to estimate the probability density function $P(Y_{NEXT}/Y_{i,j})$. The values in the PDFUNS image are generally very small with less than 1% of the image points having value greater than 0.1. As a rule of thumb, it can be argued that 1% to 0.05% of the 128x128 PDFUNS values should be greater than 0.1. A larger percentage would increase undesired randomness and noise in the synthesized image and a smaller number could cause "lock in" on the parent texture. The value 16 was also chosen for convenience and computational efficiency as it can be computed with only 4 multiplications and minimal data storage. More study could be made concerning the effect of the value of this variable on the simulation results and other approaches to creating a PDFUNS image from the MAX1 values could be tried. Studying a histogram of MAX1 values might also be very informative.

PDFUNS is then scaled so that $\sum_a \sum_b \text{PDFUNS}(a,b)=1$. In this way a pseudo-density function is formed. Finally a uniformly distributed random variable, r, [0,1] is generated and a point (c,d) is found such that

$$\sum_{a=1}^{c-1}\sum_{b} \text{PDFUNS}_{a,b} + \sum_{b=1}^{d-1} \text{PDFUNS}_{c,b} < r$$

$$\sum_{a=1}^{d}\sum_{b} \text{PDFUNS}_{a,b} + \sum_{b=1}^{d} \text{PDFUNS}_{c,b} > r \quad . \tag{7.8}$$

The Y_{NEXT} associated with the kernel shape at (c,d) is then used as the next pixel in the generated image. The process is continued until a full texture image is generated with the kernel window moving one pixel at each step, row by row.

In an indirect way, this is equivalent to generating a random variable having any distribution using the desired cumulative distribution combined with a uniformly distributed random variable (which is easy to generate). In other words, uniformly-distributed deviates are transformed to deviates having the desired distribution using the inverse cumulative density function [45,48,58]. This is frequently done in simulations.

7.4 Results

For a kernel containing 55 pixels (see Figure 7.4) passed over a 128x128 parent texture approximately 7.2×10^{6} operations (additions or subtractions) are needed to get the WMSD image defined by Eq. (7.4). Another 2.6×10^{5} are required to find the next pixel according to Eq. (7.8). therefore, to generate a 512x512 texture requires 1.96×10^{12} (2 trillion) operations.

Results from texture synthesis done by this method are shown in Figure 7.5 through 7.15. The original textures are shown in Figs. 5.1(a) through 5.11(a). Each of these images is 512x512 pixels. A 128x128 section of each original (parent) texture was used for the simulation. Bark exhibits very large macro structure and this is lost in the simulation. A similar thing happens with raffia as the kernel size is smaller than the cell size of the original texture but is not as pronounced. The top part of the bubbles texture was generated using a 128x128 portion different than that of the bottom part. For this reason the top 20-30% of the texture looks different from the rest. The large number of operations

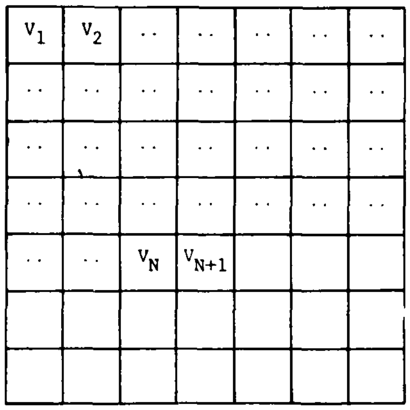


Figure 7.1 Two-dimensional Synthesis Kernel

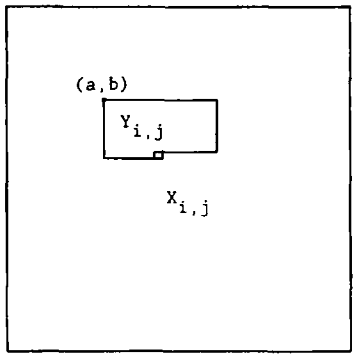


Figure 7.2 Passing Kernel Over Parent Texture

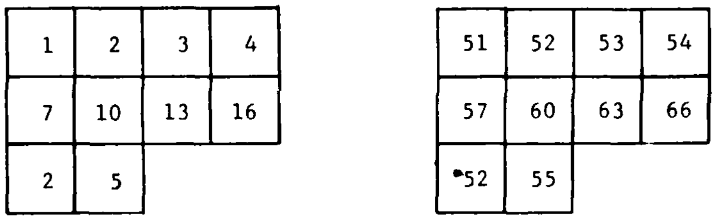


Figure 7.3 Perfectly Correlated Kernels

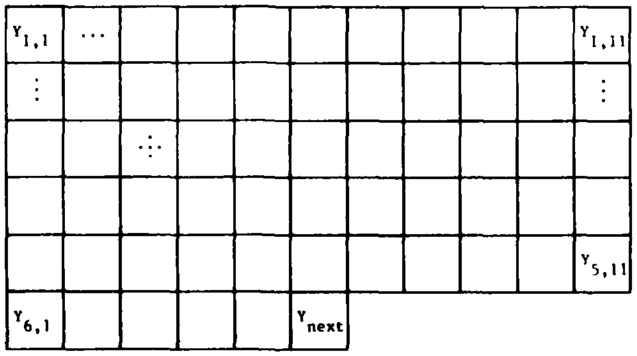


Figure 7.4 Best-fit Model Kernel

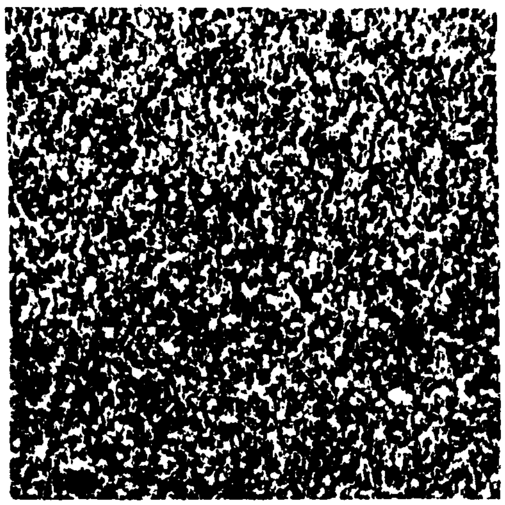


Figure 7.5 Best-fit Grass

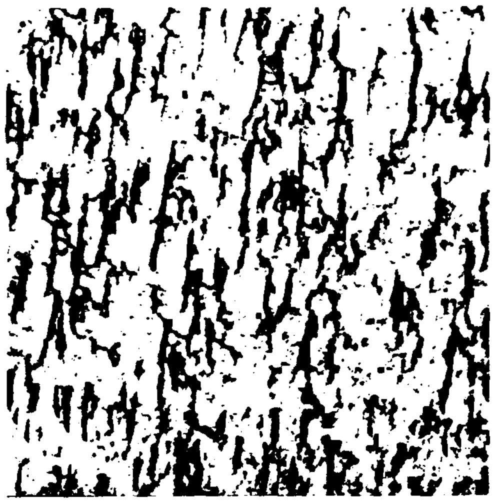


Figure 7.6 Best-fit Bark

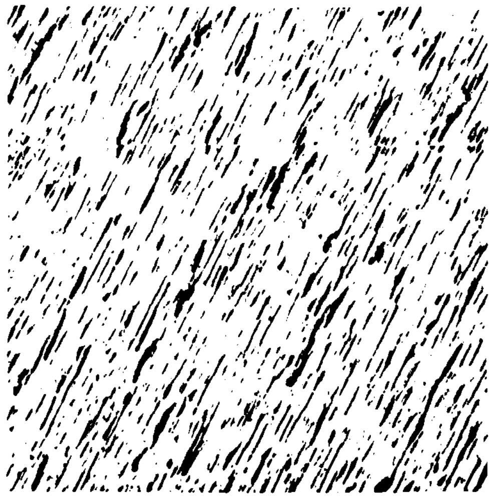


Figure 7.7 Best-fit Straw

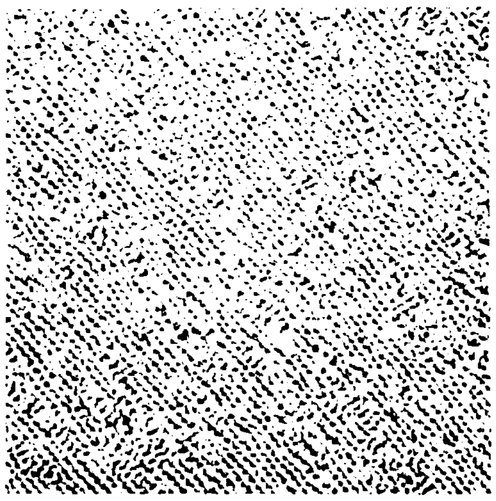


Figure 7.8 Best-fit Cloth

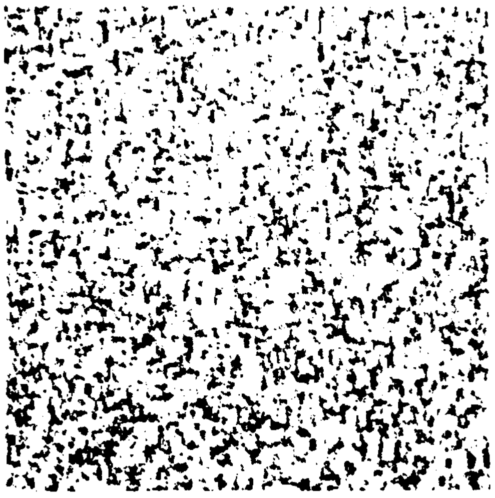


Figure 7.9 Best-fit Wool

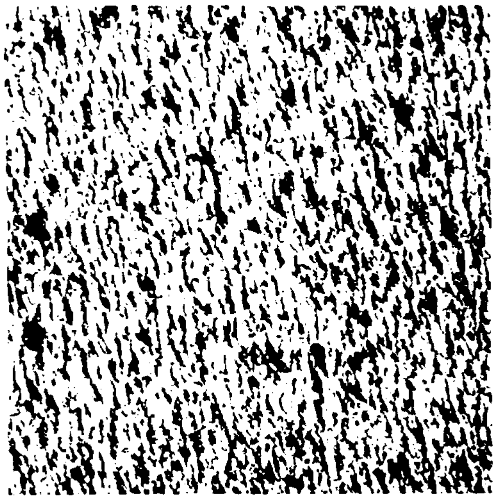


Figure 7.10 Best-fit Leather

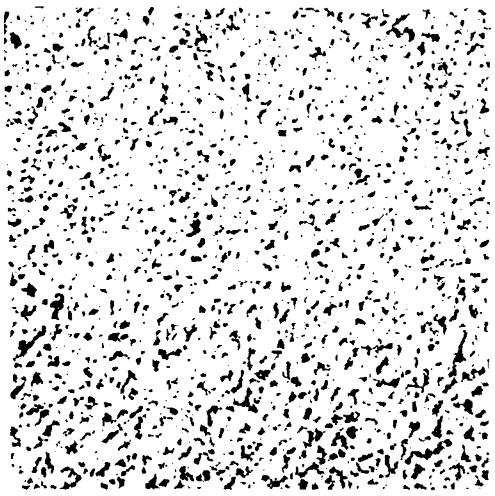


Figure 7.11 Best-fit Sand

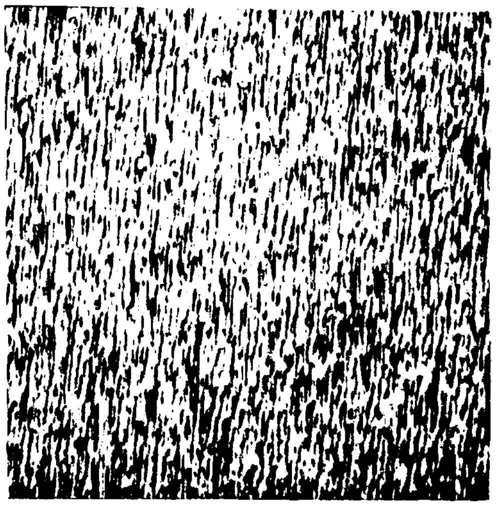


Figure 7.12 Best-fit Water

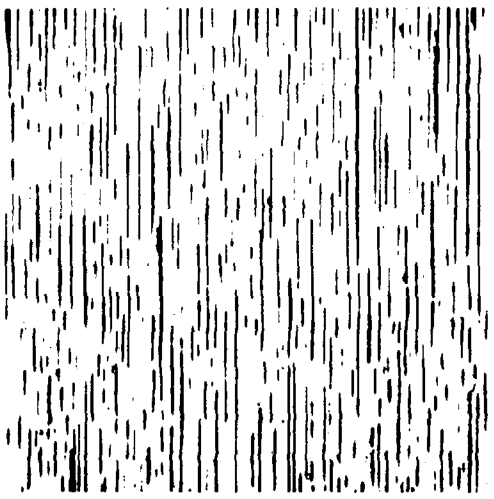


Figure 7.13 Best-fit Wood

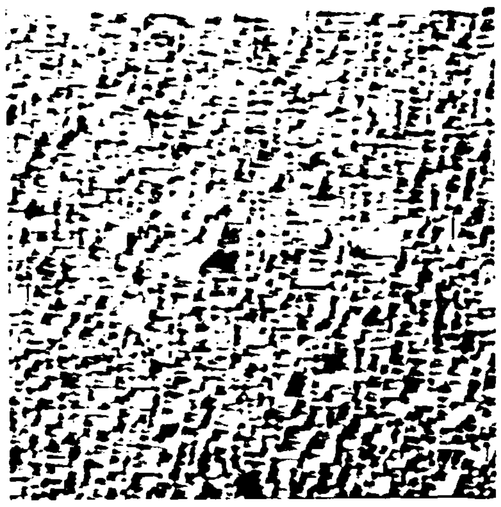


Figure 7.14 Best-fit Raffia

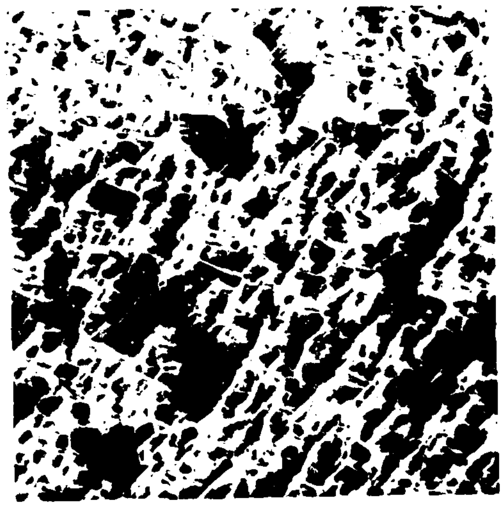


Figure 7.15 Best-fit Bubbles

makes this process very time consuming even when a pipelined processor is dedicated to the task. About 5.5 days of dedicated time on an AP120B were required to generate each texture.

Although this method is of little practical use due to the computational complexity of the algorithm a few points should be made. With constantly increasing computer processing speeds, a simplified version of this texture simulation method may be implemented in the near future. It is even possible that such computations could be performed by an array of microprocessors. In any case such brute-force approaches are simple and could be made cost-effective in the future.

The results also indicate visually the amount of texture information present in a 55 pixel window (see Figure 7.4) because at each pixel generation step, the next pixel in the Markov process depends on only the pixels in this neighborhood.

Finally, this approach is admittedly ad hoc. Numerous distance measures could replace the one chosen in this work and each would give different results that might appear better or worse. It is always important that the process be random and not merely copy the texture sample. If the simulation region is much larger than the parent sample, a deterministic process will quickly generate patterns that can easily be seen to repeat. In other words, the histogram represented by $P(V_{N+1}/V_1,\ldots,V_N)$ should rarely be a delta function. A reduction in the number of computations could be made if the kernel was non-contiguous. Also, better results could probably also be obtained if the kernel window was larger. The shape, contiguity and size of the kernel in this study was chosen primarily for computer programming considerations.

7.5 Conclusions

The results from this best-fit texture synthesis method are very pleasing but the number of computations required is large. Other similar algorithms could be developed which are simpler and could possibly produce even better results. With the decrease in computation costs and the increase in processor speeds of future computers, such texture synthesis methods could be implemented in the future without great cost.

CHAPTER 8

NON-HOMOGENEOUS TEXTURES

8.1 ·Introduction

In this chapter, methods for removing and introducing non-homogeneities in texture images are presented. Non-homogeneities in neighborhood mean and standard deviation are often removed previous to statistics collection over a parent texture image to improve the accuracy of parameter estimation. Similar non-homogeneities may be added during the texture synthesis process by merely reversing the process. In this way, synthesized textures which are homogeneous may be processed to be non-homogeneous.

8.2 Removing Non-Homogeneities

Prior to simulation attempts, the textures in this study have been preprocessed by statistical differencing [42]. This preprocessing step is described by

$$I(n_1,n_2) = [F(n_1,n_2)-\bar{F}(n_1,n_2)]\left[\frac{A\sigma_d}{A\hat{\sigma}(n_1,n_2)+\sigma_d}\right]+ \qquad (8.1)$$

$$[\alpha m_d+(1-\alpha)\bar{F}(n_1,n_2)]$$

where m_d and σ_d represent desired mean and standard deviation. F is the input pixel at location (n_1,n_2), row n_1 and column n_2 in the discrete digital image matrix, and I is the output pixel in the statistical differenced image. $\overline{F}(n_1,n_2)$ and $\hat{\sigma}(n_1,n_2)$ represent the mean and standard deviation of the input image at the point (n_1,n_2). The variable A is a gain factor that prevents overly large output values when $\hat{\sigma}$ is small, and α is a proportionality constant controlling the extent to which the mean of the output image is homogeneous.

In our studies, mean and standard deviation factors were computed in non-overlapping 16x16 pixel blocks values are used to compute the mean and standard deviation at each point. In our work $\alpha = 0.8$, $m_d = 128$, $\sigma_d = 85$ and $A = 6$. These values tend to induce local homogeneity in mean and standard deviation over an image. Large amounts of variation, however, will only be reduced and not eliminated unless A is very large and $\alpha = 1.0$.

Assuming that $\hat{\sigma}$ does not approach zero, then another form of statistical differencing can be used. This may be written in equation form as

$$I(n_1,n_2) = [F(n_1,n_2)-\bar{F}(n_1,n_2)]\left[\frac{\delta\sigma_d}{\hat{\sigma}(n_1,n_2)} + (1-\delta)\right] + [\alpha m_d+(1-\alpha)\bar{F}(n_1,n_2)] \quad . \tag{8.2}$$

Here, both α and δ are proportionality constants ranging from 0.0 to 1.0. Setting $\alpha = 1.0$ and $\delta = 1.0$ returns an output image with precise desired mean and standard deviation. Setting $\alpha = 0.0$ and $\delta = 1.0$ causes the standard deviation but not the mean of the input image to change. Setting $\alpha = 1.0$ and $\delta = 0.0$ causes the mean but not the standard deviation of the input image to be modified. Setting $\alpha = 0.0$ and $\delta = 0.0$ produces no change. Examples of statistical differencing are shown in Figs. 8.1 through 8.4. The cork texture of Fig. 8.1 is non-stationary in mean due to shading differences, primarily at the right edge. Figure 8.2 shows the image resulting from processing Fig. 8.1 using the statistical differencing algorithm. The non-homogeneity of mean is removed and the contrast is slightly increased. An original brick texture image shown in Fig. 8.3 has very low contrast. After statistical differencing, local contrast is much improved and the texture is more apparent (see Fig. 8.4). Thus the statistical differencing algorithm is quite useful in eliminating non-homogeneities.

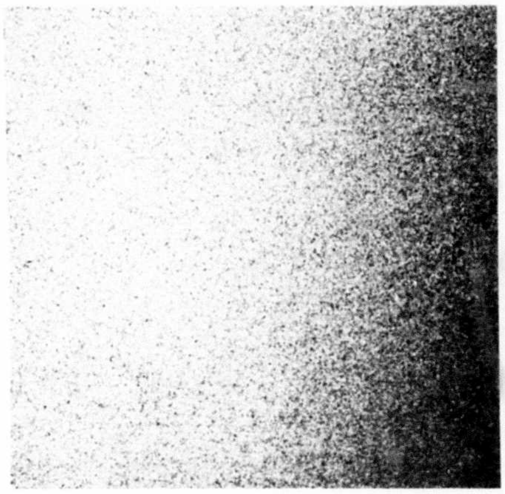


Figure 8.1 Before Statistical Differencing

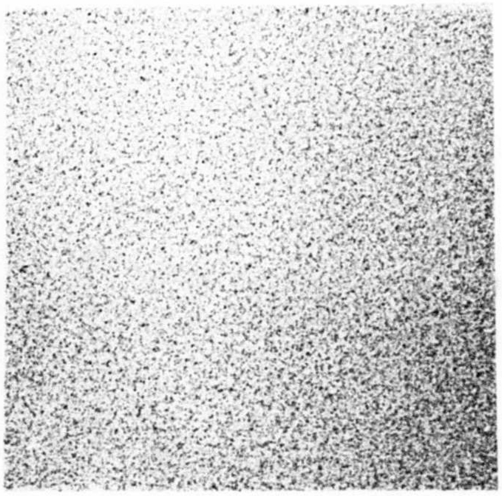


Figure 8.2 After Statistical Differencing

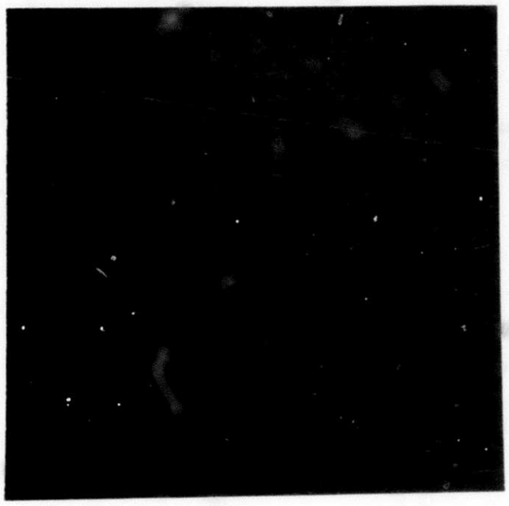


Figure 8.3 Before Statistical Differencing

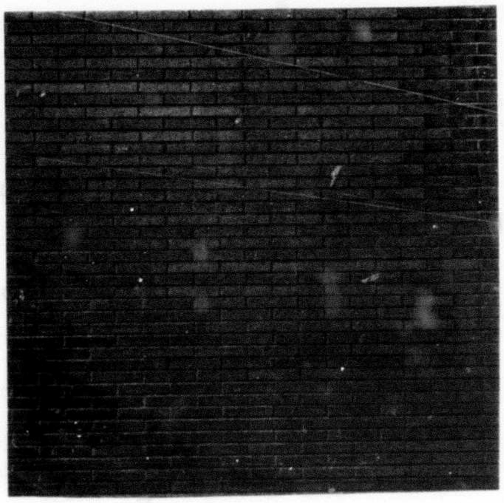


Figure 8.4 After Statistical Differencing

8.3 Introducing Non-Homogeneities

The inverse operation of statistical differencing can be called local moment modification. Solving for F in terms of I using Eq. (8.1) we find the formula for local moment modification as

$$F(n_1,n_2) = \frac{A\hat{\sigma}(n_1,n_2)+\sigma_d}{A\sigma_d} I(n_1,n_2) + \frac{\overline{F}(n_1,n_2)A\sigma_d}{A\hat{\sigma}(n_1,n_2)+\sigma_d} - [\alpha m_d + (1-\alpha)\overline{F}(n_1,n_2)] \quad . \qquad (8.3)$$

Using Eq. (8.3) we can introduce non-homogeneities into a simulated texture by generating an image $\overline{F}(n_1,n_2)$ and $\hat{\sigma}(n_1,n_2)$ and defining A, α, m_d, and σ_d.

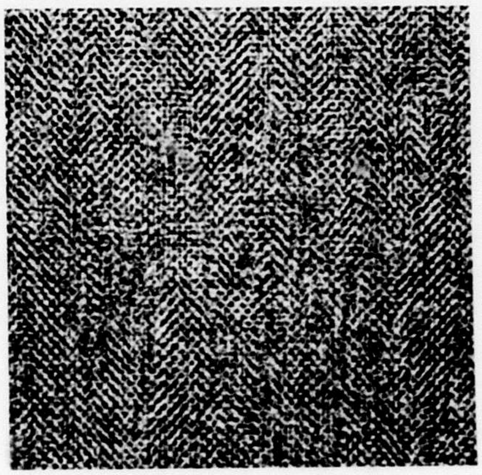
Again, if we assume that $\hat{\sigma}(n_1,n_2)$ does not approach zero, then Eq. (8.2) can be inverted to form another local modification formula

$$F(n_1,n_2) = \overline{F}(n_1,n_2) + \frac{\hat{\sigma}(n_1,n_2)[I(n_1,n_2)-\alpha m_d-(1-\alpha)\overline{F}(n_1,n_2)]}{\hat{\sigma}(n_1,n_2)(1-\delta)+\delta\sigma_d} \quad . \qquad (8.4)$$

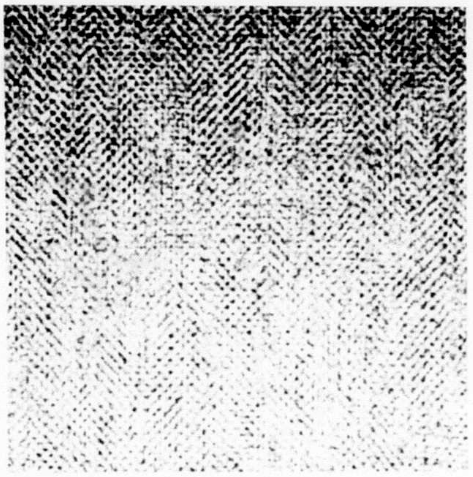
In the process of local moment modification, it is best to set m_d and σ_d to be equal to the mean and standard deviation of the homogeneous texture $I(n_1,n_2)$. Then, the mean and standard deviation of the output image $(F(n_1,n_2)$

will be defined by the images $\overline{F}(n_1,n_2)$ and $\hat{\sigma}(n_1,n_2)$. These images may be generated randomly.

An image, before and after local moment modification, is shown in Fig. 8.5. Here $\overline{F}(n_1,n_2)$ was assumed to be ramp-like and $\hat{\sigma}(n_1,n_2)$ was constant. Many other complex and random $\overline{F}(n_1,n_2)$ and $\hat{\sigma}(n_1,n_2)$ images could be used to create different effects and simulate phenomenon such as non-homogeneous lighting effects.



(a) Synthesis



(b) Synthesis After Local Moment Modification

Figure 8.5 Cloth

CHAPTER 9

TEXTURE IDENTIFICATION AND SEGMENTATION

9.1 Introduction

In this chapter we examine various approaches to texture segmentation and identification using the linear model developed earlier. These methods employ covariance measures of a texture over windows of the region being segmented or identified. These same covariance measures were used to estimate the linear model parameters which were examined in Chapter 3 and Chapter 5. In Chapter 5, the information content of these measures was shown by performing simulations of various textures and therefore, based on these results, we might conclude that these second-order statistics would be useful in the segmentation and identification of textures. Pictorial segmentation results are given in this chapter.

It is generally agreed that a great portion of texture information is contained in the second-order statistics of a texture. There are notable exceptions to this rule as was shown in Chapter 2, however for most natural textures, second-order statistics have proved to be sufficient to adequately discriminate textures in most applications [49,51]. In a practical sense, we usually are also prevented from gathering and analyzing higher-order statistics because of computational limitations. In fact, we can easily be overwhelmed by masses of data arising from second-order statistics.

The most general second-order statistics are the complete second-order joint densities, or 2-grams, which were discussed in Chapter 2 for the one-dimensional texture synthesis case. These measures may be estimated by joint gray scale histograms taken over a parent texture. For a g grey-level image, each histogram requires g^2 storage locations. But, as was pointed out in Chapter 2, there is one such histogram for each vector distance or spacing between a pair of pixels. To compute these histograms over all spacings, $2,\ldots,n_r$, in two dimensions would result in $(n_r-1)^2g^2$ entries. Even for reasonable g and n_r, this could easily create a data expansion containing unnecessary information rather than a data reduction yielding measurements with discrimination value.

For these reasons, texture image data is often quantized (to reduce g) and second-order measurements resulting in joint grey-scale histograms are made over a small number of pixel spacings (to reduce n_r). To decrease the size of the feature space further, various functions of joint grey-scale histograms are calculated and these values are primarily used to identify a texture. For a single histogram, as many as 25 to 30 functions have been proposed [7]. In spite of the large dimensionality of the feature space and the problems with quantizing low contrast textures, this family of texture features is used frequently to successfully classify textures [7,52].

Many other identification and classification schemes exist [49] as the discrimination of textures represents the most important application of texture analysis.

In this chapter, we reduce the information contained in the joint grey-scale histogram to one single number, the correlation coefficient for that particular pixel spacing. It is expected that this large reduction will cause a decrease in discrimination power as the size and information content of the feature space has been significantly reduced. The purpose of this exercise is not to develop a new, more powerful texture identifier but merely to access the information content of the correlation coefficient values when applied to the problem of texture discrimination. It is already apparent from the simulation results presented in earlier chapters that a great deal of texture information may be obtained by proper use of these correlation coefficients.

Correlation measurements have been applied in various ways to the problem of texture identification previous to this study [2,11,12,14,49,53]. It has generally been concluded that they are of lesser value than Haralick's family of functions on values of the joint grey-scale histogram [51] when applied to the discrimination problem. In the remaining sections of this chapter we will develop two new discrimination methods utilizing correlation values. One is based on the statistical test for equality of covariance matrices. The other utilizes multiple statistical tests for the equality of individual correlation coefficients. Both show good discrimination power but neither exceeds the quality of Haralick's measures.

9.2 Segmentation Using Correlation Matrices

Texture is a feature which can only be measured and identified over an area of an image. Therefore most segmentations of an image according to texture information will require that measurements be taken over an area and then part or all of that area will be classified accordingly. In our work, measurements were taken over a square, W_{BIG} pixels in length, and a center square, W_{SMALL}

pixels in length, was classified from these measurements as in Fig. 9.1.

Second-order statistics must be measured over a variety of vector distances (pixel-pair spacings) to be useful in texture discrimination. In our case, these second-order statistics are correlation values. There is only one correlation value for a particular spacing.

There are many approaches to estimating a correlation value over a window for a particular pixel-pair spacing. One involves passing a kernel of pixels over the window and taking a sample at all points where the kernel is completely contained within the window boundaries. Such measurements would result in a covariance matrix for the kernel over that window. This matrix can be used to identify a texture, as will be shown later in this section. Another approach to estimating a correlation value for a particular pixel-pair spacing would involve measurements over all possible samples within the window of that spacing. This is equivalent to passing a kernel containing two points over the window for each pixel-pair spacing. The result of this approach is a correlation value for each spacing which must be examined by itself and may not be used to form a correlation matrix as was discussed in section 3.7. A method for identifying texture using these individual correlation values will be examined in section 9.3.

Once we have obtained a covariance matrix by passing a kernel of points over a window, that matrix may be compared statistically with covariance matrices from known textures. Such a statistical test has been derived for testing the equality of covariance matrices. The maximum likelihood ratio approach used to derive this test makes the standard assumptions of multi-dimensional normality and independence of samples. Both were used in our texture simulation work. Details concerning the test and its derivation are given in [47,54]. The statistical test for two covariance matrices is given by

$$U_1 = 2.3026\, dD \cong \chi^2_{N(N+1)/2} \tag{9.1}$$

where

$$D = (M_1+M_2-2)\log_{10}|C|-(M_1-1)\log_{10}|C_1|-(M_2-1)\log_{10}|C_2|.$$

C_1 and C_2 are the estimated covariance matrices for each group,

$$C = [(M_1-1)C_1+(M_2-1)C_2]/(M_1+M_2-2)$$

and

$$d = 1 - \left[\frac{1}{(M_1-1)} + \frac{1}{(M_2-1)} - \frac{1}{(M_1+M_2-2)}\right]\left[(2N^2-3N-1)/6(N+1)\right].$$

N is the size of the covariance matrices being tested which is equal to the number of pixels in the kernel. M_1 and M_2 are the sample sizes used to estimate C_1 and C_2. $|C|$ denotes the determinant of C. The test statistic, U_1, has an approximate chi-square distribution with $N(N+1)/2$ degrees of freedom and approaches 0 as C_1 approaches C_2.

Having derived the test for equality of two covariance matrices we must determine the contents of these matrices. As the number of points in the kernel increases, the size of the covariance matrices increases leading to more difficult and time-consuming determinant calculation required in Eq. (9.1). Also, as the spacing of these pixels increases, the number of samples in a window decreases. Finally, as more and more points are included in the kernel, the amount of redundant and overlapping information in the covariance matrix increases due to redundant pixel-pair spacing. This was discussed in section 3.7.

To eliminate this redundancy, we will consider the problem briefly in one-dimension. Perfect, non-redundant pixel spacing is possible only for patterns with maximum range of 1, 3 or 6 pixels given by the corresponding patterns XX, XX-X and XX--X-X as shown in Table 9.1. For these three particular ranges, the patterns shown are constructed so that no two pairs of X's are separated by the same distance. One and only one pair of X's may be found in these patterns which is separated by each distance less than or equal to the maximum range. For all other ranges, redundancies will exist in any pattern spanning the range. That is, more than one pair of X's may be found which are separated by the same distance in patterns not having a maximum range of 1, 3 or 6. A pattern spans a range if and only if at least two X's may be found in the pattern which are separated by every distance less than or equal to the maximum range. These patterns are sometimes referred to as difference sets [59,61,73].

A set containing a minimal number of X's can be found to span all ranges. A list containing the minimum number of X's required to span each range and one non-unique pattern which spans the range is given in Table 9.1. (To determine that the distance 52 could not be spanned with 12 points required over 132 billion subtractions plus a very large number of logical operations.)

Extending these one-dimensional spacings to two dimensions merely requires the vector product of the transpose of any of the row vector patterns with itself yielding a two dimensional matrix. Unfortunately, the corresponding two-dimensional matrix pattern will always contain redundancies if all possible distances and orientations less than or equal to the maximum range are to be spanned. Patterns obtained using this vector product approach are heavily weighted in the horozontal and vertical directions.

A two-dimensional kernel containing 16 pixels which spans a two-dimensional range of 6 is shown in Fig. 9.2. Passing this kernel over a window yields a covariance matrix of dimension 16, which is not an unreasonable size for computation purposes. The number of data samples over a window of size W_{BIG} is $(W_{BIG}-6)^2$. Even for small windows of size 16, a reasonable sample size of 100 may be obtained.

Once the kernel and procedure are determined, we proceed with the process of segmenting the textures to test the identification procedure. There are many approaches to this problem which are usually defined by the particular case of interest. We may or may not have prototype and parent texture data. We can segment, cluster-analyze or identify. Notice that the generality of the test leaves a wide variety of options open. We can compare matrices in one area of an image with those in other areas and our test statistic values will be the distance measures defining the closeness of the textured

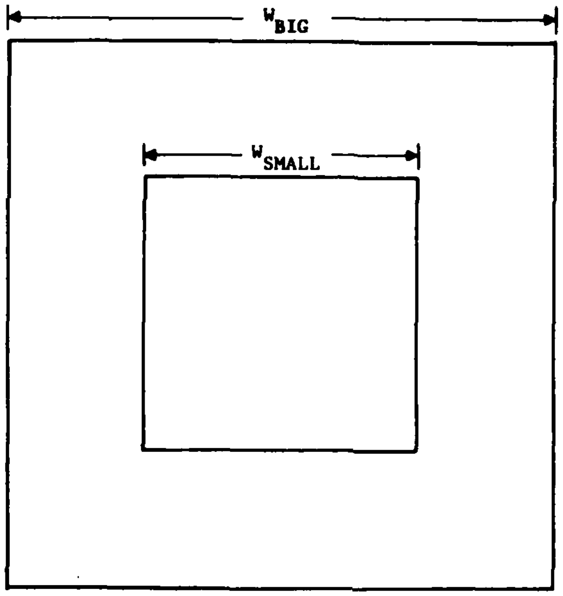


Figure 9.1 Segmentation Window

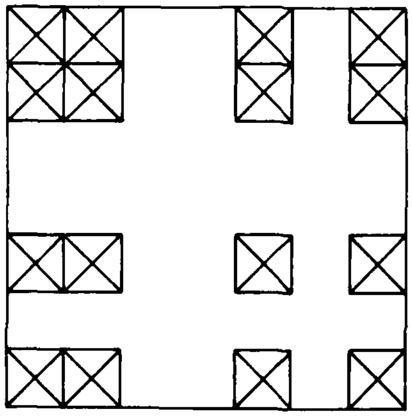


Figure 9.2 Two-dimensional Spanning Kernel

Table 9.1 MINIMAL SPANNING SETS

RANGE	#PTS	
2	2	XX
3	3	XXX
4	3	XX-X
5	4	XXX-X
6	4	XXX--X
7	4	XX--X-X
8	5	XXXX---X
9	5	XXX--X--X
10	5	XXX---X--X
11	6	XXXX--X---X
12	6	XXXX---X---X
13	6	XXXX----X---X
14	6	XXX---X---X--X
15	7	XXXXX----X----X
16	7	XXXXX-----X----X
17	7	XXXX----X---X---X
18	7	XXXX----X----X---X
19	8	XXXXXX------X-----X
20	8	XXXXX----X----X----X
21	8	XXXXX-----X----X----X
22	8	XXXXX-----X-----X----X
23	8	XXXX----X----X----X---X
24	8	XXX--------X---X--X--X-X
25	9	XXXXXX------X-----X-----X
26	9	XXXXXX------X------X-----X
27	9	XXXXX-----X----X-----X----X
28	9	XXXXX-----X-----X-----X----X
29	9	XXX---------X----X--X--X--XX
30	9	XXX----------X---X--X--X--X-X
31	10	XXXXXX-----X------X-----X-----X
32	10	XXXXXX------X-----X------X-----X
33	10	XXXXXX------X------X------X-----X
34	10	XXXXX-----X-----X-----X-----X----X
35	10	XXXX-----------X----X---X---X--X--X
36	10	XXX--------------X---X--X--X--X--X-X
37	10	XX-X--X------X------X------X---X---XX
38	11	XXXXXXX-------X-------X-------X------X
39	11	XXXXXX------X------X-----X------X-----X
40	11	XXXXXX------X------X------X------X-----X
41	11	XXXXX-----X------X------X----X-----X----X
42	11	XXXX----------X----X------X----X---X---X-X
43	11	XXXX---------------X----X---X---X---X--X--X
44	11	XX-X--X------X------X------X------X---X---XX
45	12	XXXXXXX-------X-------X------X-------X------X
46	12	XXXXXXX-------X-------X-------X-------X------X
47	12	XXXXXX------X------X------X------X------X-----X
48	12	XXXXXX------X-------X-------X-----X------X-----X
49	12	XXXX---------X------------X--X---X-----X---X---XX
50	12	XXXX--------------------X----XX---X---X---X---X--X
51	12	XXXX-------------------X----X---X---X---X---X--X--X
52	13	XXXXXXXX--------X--------X--------X--------X-------X
53	13	XXXXXXX-------X------X--------X------X-------X------X
54	13	XXXXXXX-------X-------X-------X-------X-------X------X
55	13	XXXXXXX-------X--------X--------X------X-------X------X
56	13	XXXXXX------X-------X-------X-------X-----X------X-----X

region. We could compare a matrix with a number of known prototypes and classify the unknown region according to the test statistic for each comparison. Finally, we may merely wish to locate a particular texture in an image and thus we would compare matrices with only one prototype.

This later approach was selected in this chapter as the visual results are simple to display and analyze. We will compare the matrix measured over a large window with one obtained over the complete parent texture prototype and assign the pixels in the small window accordingly (see Fig. 9.1). Figure 9.3 shows the test composite image used. We will attempt to identify the texture sand in that region. The texture sand was chosen as the simulation results of this texture were in some ways very poor when examined critically on a high resolution device. Therefore, it represents a worst-case example.

Figure 9.4 shows the segmentation results when $W_{BIG} = W_{SMALL} = 16$. The pixel brightness are linearly mapped test statistic values, which are supposed to be chi-square-distributed. Improvement is made when $W_{BIG} = 32$ and $W_{SMALL} = 16$ as is seen in Fig. 9.5. Figure 9.6 shows much improved results when $W_{BIG} = 48$ and $W_{SMALL} = 16$. The difference is due to the availability of more texture information in the larger window in the form

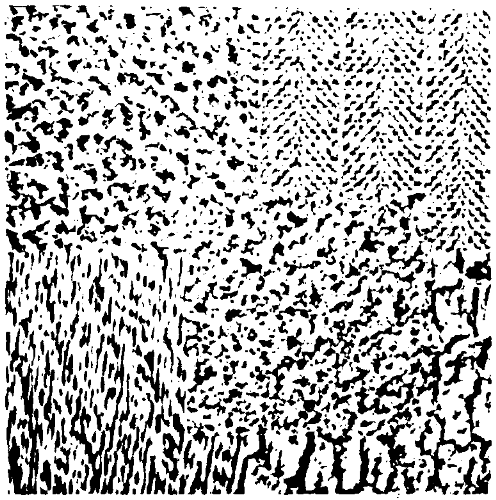


Figure 9.3 Input Composite Image

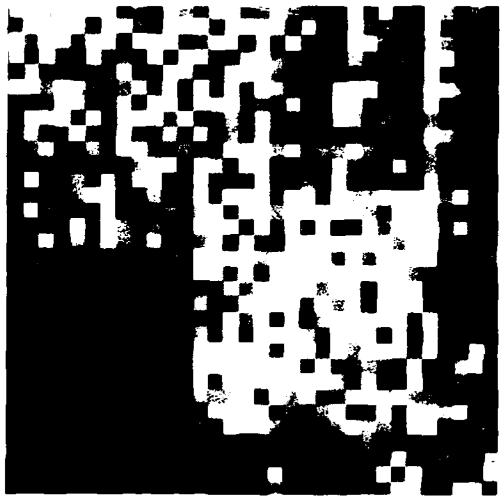


Figure 9.4 Segmentation Using Covariance Matrix, $W_{BIG} = 16$

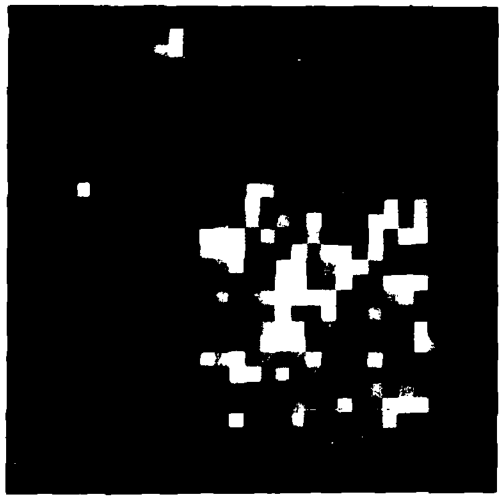


Figure 9.5 Segmentation Using Covariance Matrix, $W_{BIG} = 32$

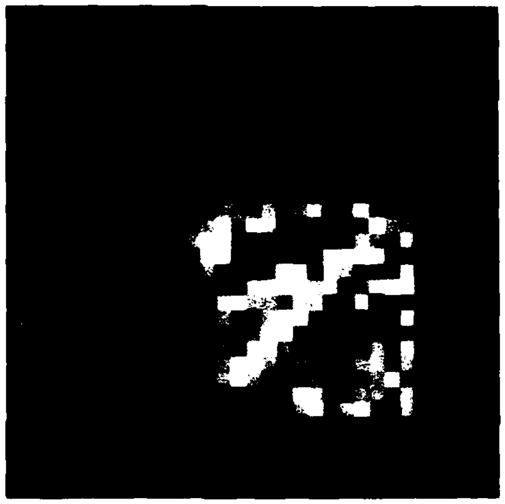


Figure 9.6 Segmentation Using Covariance Matrix, $W_{BIG} = 48$

of increased sample size (100 to 1764).

As a final note, it should be understood that this chosen texture kernel (see Figure 9.2) totally ignores information obtained earlier concerning the interdependence of pixels in the generating kernel as expressed in the linear model. It is very possible that better segmentation results could be obtained by using the kernel shape which best fits each texture as this particular pattern of points was found to be most significant. The linear model used for simulation of each texture might even be used itself as the distributions of the parameter estimates are known if certain assumptions are made [24,39]. But neither of these approaches have been tried even though improvements are expected.

9.3 Segmentation Using Individual Correlation Coefficients

Individual correlation estimates for particular pixel-pair spacings may be made by taking measurements over all possible samples within the window containing that spacing. This approach utilizes more information than the fixed kernel method as it includes measurements near all edges of the window (thus, the sample size is increased). The result is a single coefficient for each pixel-pair spacing. If the spacing between each pair is unique then the set of coefficients will be non-redundant. The covariance matrices used to segment textures in the last section may contain redundant information if the pixel-pair spacings in the kernel are repeated.

A statistical test for comparison of two correlation coefficients has been derived for the bivariate normal distribution and is

$$U_2 \overset{\vee}{=} \chi_1^2 \qquad (9.2)$$

where

$$U_2 = (Z_1-\bar{Z})^2(M_1-3)+(Z_2-\bar{Z})^2(M_2-3)$$

and

$$Z_1 = \text{arctanh}\ (r_1)$$
$$Z_2 = \text{arctanh}\ (r_2)$$
$$Z_3 = [(M_1-3)Z_1+(M_2-3)Z_2]/(M_1+M_2-6)$$

r_1 and r_2 are the computed correlation coefficients from the two groups. This test is derived using the fact that Z_1 is approximately distributed $N(Z_1{:}\ \text{arctanh}(\rho_1), (M_1-3)^{-1})$ where ρ_1 is the true correlation coefficient of the population (see section 3.5). For additional details see [39]. Again, to derive the tests assumptions of normality and independent samples are made.

As in the previous section, the test statistics may be used to segment or identify textures. However, difficulty arises as our test must be performed for each measured correlation value. Thus a number of U_2 values are obtained if multiple pixel-pair spacings are used. At this point these values can be combined in numerous ways. As each U_2 value is chi-square distributed we can compute the probability, p_{U_2}, that a random variable, which has that chi-square distribution, is greater than or equal to U_2 as shown in Fig. 9.7. For $U_2 = 0$ $(r_1 = r_2)$ this probability is 1.0. A product of the probabilities associated with each U_2 was used to indicate the overall fit of one set of correlation coefficients to another set.

Finally, it should be noted that tests for equality of correlation coefficients will not detect differences in mean and variance over the window. The correlation coefficient specifically removes these effects. As a result, it may be advantageous to include tests for equality of means and variances into the comparison process. For equality of means the test statistic is

$$t_1 = \frac{\hat{\mu}_1 - \hat{\mu}_2}{\sqrt{\frac{(M_1-1)\hat{\sigma}_1^2+(M_2-1)\hat{\sigma}_2^2}{M_1 + M_2 - 2}}\sqrt{\frac{1}{M_1} + \frac{1}{M_2}}} \cong t_{M_1+M_2-2} \tag{9.3}$$

where $\hat{\mu}_1$ and $\hat{\mu}_2$ are the calculated mean and $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$ are the calculated variances for the two groups being compared. This test statistic is a value of a random variable having the t-distribution with M_1+M_2-2 degrees of freedom. For the equality of variances the test statistic is

$$t_1 = \frac{\hat{\sigma}_1^2}{\hat{\sigma}_2^2} \cong F_{M_1-1,M_2-1} \quad . \qquad (9.4)$$

This test statistic is a value of random variable having an F-distribution with M_1-1 and M_2-1 degrees of freedom. The derivation of both of these tests requires the assumption of normality for the two groups and an independent sample set. The known distributions of the test statistics t_1 and f_1 was used to obtain the probabilities, p_{t_1} and p_{f_1} that a random variable having that distribution would be greater than or equal to the calculated values of $|t_1|$ and $MAX(f_1,1/f_1)$ or less than $-|t_1|$ and $MIN(f_1,1/f_1)$ (see Figs. 9.8 and 9.9).

Having obtained the set of probabilities, p_{U_2}, and p_{t_1} and p_{f_1} corresponding to the set of tests for the equality of correlation coefficients and the tests for equality of mean and variance of the window a single test statistic

$$\sum_i \log(MAX(p_{U_{2_i}},10^{-6}))+\log p_{t_1}+\log p_{f_1} \qquad (9.5)$$

is formed to indicate the combined fit of these tests. The probabilities are log-scaled to eliminate scaling

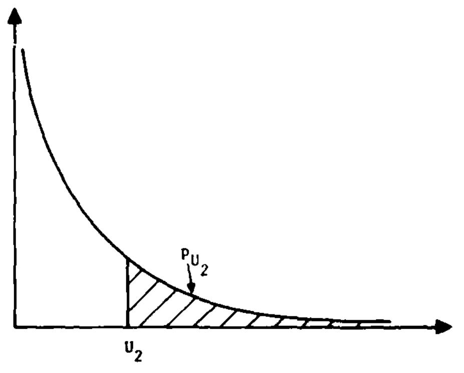


Figure 9.7 The Chi-square Distribution

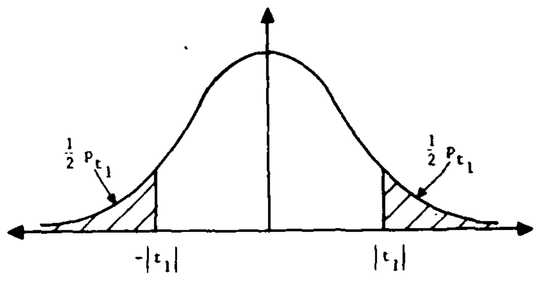


Figure 9.8 Student's t-Distribution

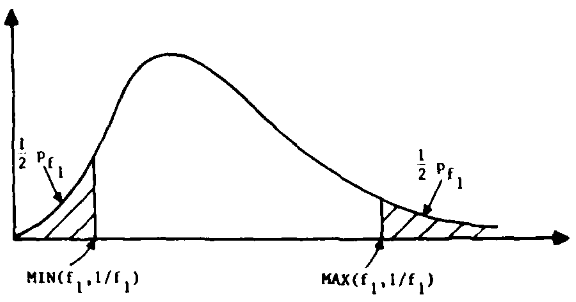


Figure 9.9 The F-Distribution

problems which would occur when taking the product of many small numbers. The $p_{U_{2_i}}$'s are bounded below to reduce the effect of single, noisy correlation coefficients. These output test statistics obtained by comparing a windowed region of Fig. 9.3 with statistics gathered from the parent prototype texture sand were linearly scaled to produce the intensity images shown in Figs. 9.11 to 9.13. The results for $W_{BIG} = W_{SMALL} = 16$ are shown in Fig. 9.11. Improved results for $W_{BIG} = 32$ and $W_{SMALL} = 16$ are shown in Fig. 9.12. Results for $W_{BIG} = 48$ and $W_{SMALL} = 16$ are shown in Fig. 9.13.

9.4 Conclusions

In implementing the two procedures detailed in this chapter, large values for the test statistics U_1 and U_2 were obtained even when comparing the matrices or correlation values measured from an extracted portion of a parent texture with those obtained from the entire parent texture. There are two basic reasons for this. First, the assumptions of normality are probably incorrect. There is little reason to believe that the distribution of pixels in an image is truly multi-variate normal. Secondly, the samples used to estimate the covariance matrix and the correlation coefficients are not random and independent. They are strongly related by their spatial

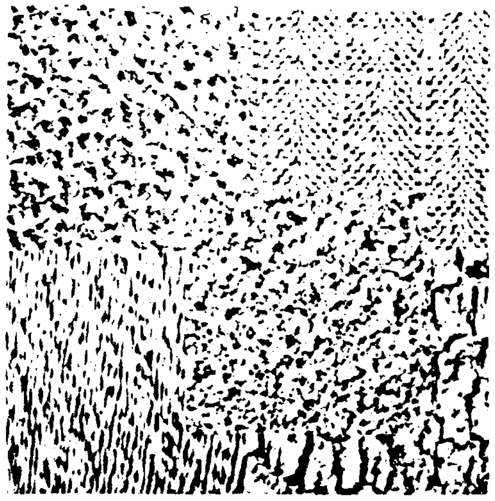


Figure 9.10 Input Composite Image

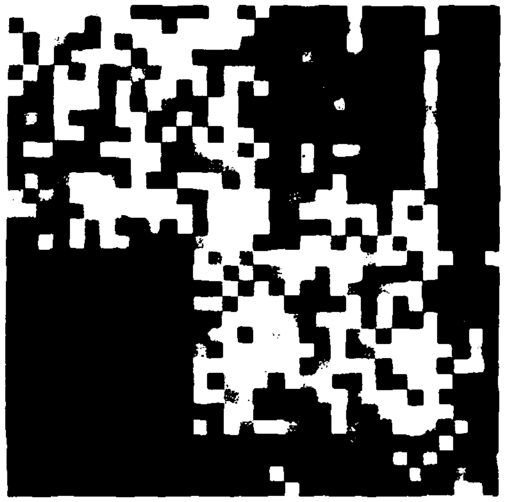


Figure 9.11 Segmentation Using Individual Correlation Values, $W_{BIG} = 16$

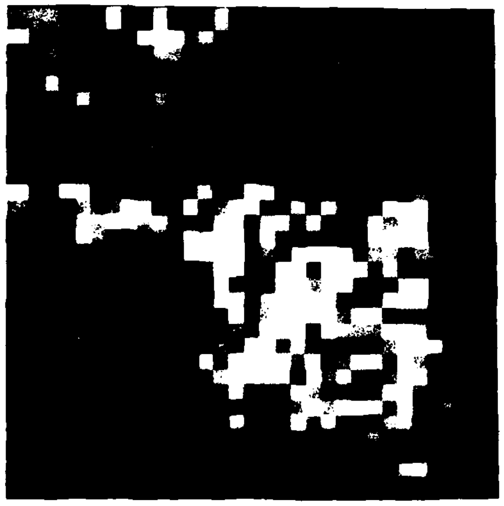


Figure 9.12 Segmentation Using Individual Correlation Values, $W_{BIG} = 32$

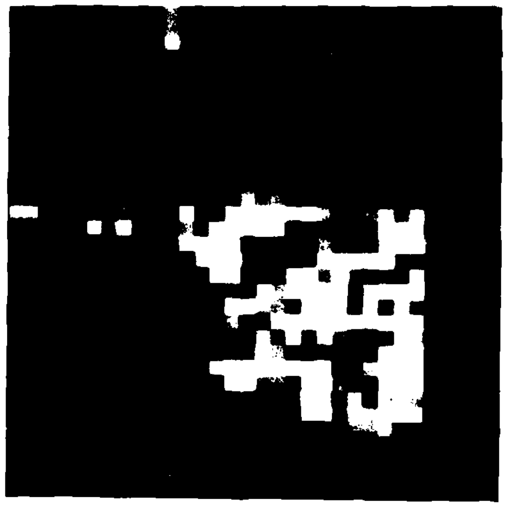


Figure 9.13 Segmentation Using Individual Correlation Values, $W_{BIG} = 48$

positions in the image and are often adjacent. The sample size may be adjusted to reflect this fact but no work in this area was done.

In most cases, statistical tests may be considered to be robust - not weakened significantly - when assumptions used to derive them are violated. This is probably not true of the test for equality of two covariance matrices which has been called "frail at best" [47]. For these reasons, the methods presented may be considered innovative but not necessarily statistically sound.

In spite of this fact, the results show that the method using the test for equality of covariance matrices was superior to the method involving multiple tests of individual correlation coefficients. This could be due to the method used to combine the multiple tests.

The pictorial results of this chapter indicate the usefulness of correlation values in texture identification. The methods are not intended to be improvements over existing segmentation identification techniques. An adequate number of discrimination techniques have been proposed by researchers already. The discussion is intended to apply a specific texture synthesis model which was based on second-order statistics to the texture identification problem and this was done successfully.

CHAPTER 10

SUMMARY

10.1 Introduction

In this chapter, the results of each of the texture synthesis methods presented in this thesis are discussed and compared. This summary will clarify the methods and analyze the favorable and unfavorable characteristics of each.

10.2 Tabulation and Discussion of All Models

A complete synopsis of the synthesis methods presented in this thesis is effectively contained in Table 10.1. Listed in this table are the eleven methods of texture generation and simulation presented in Chapter 2 through Chapter 7. The first three columns of the table state the location of the text and figures (output) associated with each and also the key figure or equation number which identifies the method in a simple, straightforward manner.

The next three columns are used to evaluate the complexity of statistics collection, statistics storage

Table 10.1 SUMMARY AND COMPARISON OF TEXTURE SYNTHESIS METHODS

Method	Text Location	Figure Output	Key Equation or Figure	Computational Requirements For Statistics Collection	Storage Requirements For Statistics	Computational Requirements For Generation Process	Quality Of Simulation	Comments
One-Dimensional Binary Model	2.1-2.8	2.5-2.12	Fig. 3.1(a)	-	16	2 mins.	-	not useful for natural texture simulation
Binary N-Gram Model	3.1-3.8	3.5(b)-3.14(b)	Eq. 3.2,3.3	6 hours	65,536	2 mins.	7	storage consuming
Extended Binary N-Gram Model	3.9	3.15	-	7 hours	20,000	8 mins.	8	extension improves structure
Binary Autoregressive Model	3.10	3.5(c)-3.14(c)	Eq. 3.22	6 hours	120	5 mins.	6	low storage - good results
Algebraic Reconstruction Model	4.1-4.4	4.1-4.2	Eq. 4.6	16 hours	65,536	2 mins.	5	high computation cost
Autoregressive Linear Model	5.1-5.6	5.1(b)-5.11(b)	Eq. 5.4	6 hours	120	5 mins.	6	good data compression and results
Second-Order Linear Model	5.7	5.1(c)-5.11(c)	Eq. 5.45 Fig. 5.13	25 hours	250	10 mins.	7	slightly better than first-order but probably not worth the extra effort
Second-Order Linear Model With Non-Stationary Noise	5.8	5.1(d)-5.11(d)	Figs. 5.16,5.17	25 hours	8,250	12 mins.	8	non-stationary noise improves results at little storage expense
Skip-Generate Model	6.2	6.5-6.12	Figs. 6.1-6.4	4 hours	960	8 mins.	6	simple data collection process may be good on textures with large structure
Piecewise-Linear Autoregressive Model	6.3	6.17	Figs. 6.13,6.14	12 hours	960	6 mins.	6	probably not worth added complexity of model
Field-Definition Model	6.4	6.20,6.23	Fig. 6.15	15 hours	250	6 mins.	8	excellent for textures composed of more than one subtexture
Best-Fit Model	7.1-7.4	7.5-7.15	Eq. 7.4 Fig. 7.2	0 mins.	16,384	20 days	10	excellent results brute-force approach

and the generating process as presented in Fig. 1.1. The computational requirements are approximate and are indicated in CPU time of a DEC KL-10. Naturally, these numbers are relative to the processor used. The storage requirements are given in terms of full-word processor locations needed for statistics storage. In some cases, this storage could be reduced by packing more than one number (especially integers) into one 32- or 36-bit word but that was not done here. For example, four 8-bit integer values will fit into one 32- or 36-bit word.

The seventh column provides a relative measure of the quality of the texture simulation on a zero to ten scale. A value of 5 indicates a good or reasonable simulation. As synthesis evaluation is a nebulous process so is the assignment of relative merit. The assessment of results is internal to this thesis as there is little synthesis work in the general literature.

The last column of the table contains important general comments on each method.

The one-dimensional binary generation method of Chapter 2 permitted study of visual response to changes in the probability distribution of texture. Although the method was not useful for natural texture simulation, it did lead to the models of later chapters.

The N-gram model of Chapter 3 was an extension of the Chapter 2 model applied to two-dimensional natural textures. The results were very good even with the severe constraint imposed by the upper limit on the number of pixels allowed in the generation kernel. With an increase in the complexity of the collection process this model was extended (see section 3.9) to allow both a larger number of pixels to be in the kernel and an increase in kernel range. The extension produced better simulations of structured textures. Finally, in section 3.10, the binary linear model, which was used to determine the contents (shape) of the generation kernel, was used to generate binary textures. The textures generated using this model were nearly equal in quality to those of the more complex and storage-consuming N-gram model. The N-gram model of Chapter 3 uses a generation kernel whose contents (shape) depends on the linear model. Therefore, the number of computations required in the statistics collection portion of the N-gram model necessarily includes computations of the linear model. However, in some cases, points which lie far from the kernel eye can be neglected in the N-gram model as only the best few are used due to storage limitations. On the other hand, such points should be included in the linear model therefore a larger neighborhood surrounding the kernel eye should be used in the estimation of the linear model.

Realizing the power of second-order statistics, a method of reproducing Nth order statistics using algebraic reconstruction was presented in Chapter 4. The approach proved to be academic as the number of iterations leading to a solution yielding adequate synthesis results was very large and required much storage and computation. The simulations were also slightly less appealing than the methods of Chapter 3.

In Chapter 5, the linear autoregressive model of Chapter 3 was applied to 256-gray-level imagery. The results were good considering the vast reduction of information caused by the statistics collection process. Slightly better results were obtained by allowing the model to contain cross terms but the resulting complexity suggests that the change in texture quality is not worth the added effort and computational expense. Using non-gaussian, non-stationary noise in the model (see section 5.7) produced markedly better results but with a requirement of slightly increased storage.

The skip-generate method of Chapter 6 may be used to improve the simulation of textures having a coarse structure. The model produces results equal in quality to the linear autoregressive model of Chapter 5 while requiring fewer computations in the collection process. The piecewise linear autoregressive model presented later in Chapter 6 promises an analytically superior fit but produced results which were not appealing enough to warrant additional effort.

The field-definition model presented at the end of Chapter 6 is useful when generating textures composed of subtextures. The idea of defining fields for texture generation is a powerful approach for both statistics collection and texture synthesis but the slow response of the autoregressive model to boundaries produced results less appealing than expected.

The best-fit model of Chapter 7 represents a brute-force approach to texture synthesis. Though computationally demanding, the final results show that excellent texture simulations can be generated using complete statistics from a relatively small neighborhood. The problems with a small neighborhood are seen in the simulation of regular textures such as raffia where the size of the primitives in the texture is much greater than the window used in the best-fit calculation.

Combining the method of choosing a kernel shape using a linear model (discussed in Chapter 3) with the skip-generate and best-fit models would result in a very powerful, but extremely computationally-demanding texture synthesis method. The effort required to generate textures by such a complex combination of methods would not be required in many cases where simpler models will produce adequate results. The complexity of the model depends on the texture being simulated.

The ideas presented in Chapter 8 allow the introduction and removal of texture non-homogeneities. In Chapter 9, texture identification methods using the statistics employed in model parameter calculation of earlier chapters were proposed. Although they do not discriminate textures as well as the methods of other researchers they do illustrate the application of synthesis models to texture identification problems.

10.3 Suggestions for Future Study

Many approaches to texture analysis have been carried out in the frequency domain by using transform and filtering techniques. Texture synthesis could be carried out in such a domain or on frequency-filtered image. For example, numerous textures could be generated in non-overlapping frequency planes and then added together to obtain a final texture synthesis. However, each of these planes is probably interdependent and a simple generation with summation is probably not possible.

Another future approach might employ to a greater extent those statistics useful for texture identification and discrimination. In most cases, these measurements are not readily suitable to a synthesis process but with careful study, many could possibly be used in such a manner. Still, there is little evidence thus far to indicate that statistics useful for texture identification will be useful for texture synthesis.

An area which deserves immediate attention involves preprocessing of texture by convolution. Noise filtering and smoothing could be useful in improving model parameter estimates. Also, a deconvolution processing of images synthesized to resemble convolved textures might produce excellent simulation results.

Along this line, another synthesis method should be considered. With any model there is always unexplained variance which the model fails to account for when it is applied to the original parent texture. Ideally, this unexplained variance would be merely noise but this is rarely the case in practice. It is possible to develop additional models which could be used to explain (or simulate) this previously-unexplained variance. Combining these new models with the original model would result in an improved synthesis method.

Finally, texture is often produced by the shading effect of a light source on a three-dimensional object. Therefore, to generate a texture, a three-dimensional relief could be formed then shaded using current, commonly-used graphics techniques. Such an approach could be worthwhile in some cases but in most, synthesizing three-dimensional relief (which is the same as generating height information over a two-dimensional grid) is equivalent to generating intensity information over a two-dimensional grid.

10.4 Conclusion

Many natural textures are generated using a variety of methods presented in this thesis. The quality of the natural texture simulations depends on the amount of computation and storage used in each process. Many textures were adequately simulated using simple models thus providing a potentially great data compression for many applications. Others required more extensive computation to synthesize visually pleasing results. Thus, as might be expected, the success of any synthesis method is highly dependent on the texture itself. When examining the results of any method the characteristics of both the model and the textures used must be considered.

It would be unwise to believe that all textures could be generated using any single approach, especially one which promises to compress texture information to a handful of numbers. Yet this is precisely what has been attempted in the texture synthesis work of this thesis.

It is important to note the power and complexity of each synthesis method of this thesis. Many textures can be simulated well using simple models such as the autoregressive model if the model is carefully constructed. Improvements in texture simulation were made by modifying these models and allowing them to become more complex and use more information in the generation process. Other textures require more complex models such as the best-fit model of Chapter 7. The shortcomings of each method will constantly indicate where future work can be done.

APPENDICES

APPENDIX A

GENERATING BINARY TEXTURE PAIRS POSSESSING EQUAL SECOND-ORDER STATISTICS

Texture pairs may be generated which have equal second-order statistics but are visually discriminable. Let $G_a(V_1,V_2,V_3,V_4)$ represent the probability of generating a 0 after the pixels V_1,V_2,V_3,V_4 along a line in a one-dimensional texture (a) and $G_b(V_1,V_2,V_3,V_4)$ in textures (b). (This is a slight change of notation from Chapter 2 as (a) and (b) will be texture number indices in this Appendix.) Define

$$V_i' = |1-V_i| \qquad (A.1)$$

where $V_i \varepsilon \{0,1\}$.

The restrictions used to generate textures with equal second-order statistics, $P_a(V_1,V_j) = P_b(V_1,V_j)$, in Chapter 2 may be stated as

$$G_a(V_1,V_2,V_3,V_4) = G_a(V_1',V_2,V_3',V_4') \qquad (A.2)$$

$$G_a(V_1,V_2,V_3,V_4) = 1-G_a(V_1,V_2',V_3',V_4) \qquad (A.3)$$

and

$$G_a(V_1,V_2,V_3,V_4) = G_b(V_1,V_2,V_3',V_4) \quad . \tag{A.4}$$

Equations (A.2) and (A.3) must also hold for texture (b). Combining Eq. (2.14) and (2.24) for a 4-gram system, it can be shown that

$$P_a(V_1,V_2,V_3,V_4) = P_a(V_1,V_2',V_3',V_4) \tag{A.5}$$

$$P_a(V_1,V_2,V_3,V_4) = P_b(V_1,V_2,V_3',V_4) \quad . \tag{A.6}$$

Combining the above restrictions yields two further rules,

$$[V_5+(-1)^{V_5}G_a(V_1,V_2,V_3,V_4)]=[V_5'+(-1)^{V_5'}G_b(V_1,V_2',V_3,V_4)] \tag{A.7}$$

$$\begin{aligned}[V_5+(-1)^{V_5}G_a(V_1,V_2,V_3,V_4)]&=[V_5'+(-1)^{V_5'}G_a(V_1,V_2',V_3',V_4)]\\ &=[V_5'+(-1)^{V_5'}G_a(V_1',V_2',V_3,V_4')] .\end{aligned} \tag{A.8}$$

These restrictions yield textures having the equalities

$$P_a(0)= P_b(0) = P_a(1) = P_b(1) \, . \tag{A.9}$$

As no closed-form solutions to Eq. (2.14) and Eq. (2.24) are easily obtained the proofs verifying Eq. (A.5) and (A.9) are very complex. However, these properties may be illustrated by generating numerous examples where Eq. (A.2), Eq. (A.3) and (A.4) hold.

Assuming the above conditions, a proof then follows to show that $P_a(V_1,V_j) = P_b(V_1,V_j)$ if the above restrictions hold:

$$P_a(V_1,V_j) = \sum_{\substack{V_i \\ i \neq i,j}} P_a(V_1,V_2,\ldots,V_{j-1},V_j)$$

$$= \sum P_a(V_1,V_2,V_3,V_4) \prod_{k=5}^{j} [V_k + (-1)^{V_k} G_a(V_{k-4},V_{k-3},V_{k-2},V_{k-1})]$$

$$= \sum P_b(V_1,V_2,V_3',V_4) \prod_{k=5}^{j} [V_k' + (-1)^{V_5'} G_b(V_{k-4},V_{k-3}',V_{k-2},V_{k-1})]$$

$$= \sum P_b(V_1,V_2',V_3,V_4) \ . \tag{A.10}$$

$$\prod_{k=5}^{\left[\frac{j-4}{3}\right]} \Big\{ [V_{3k-2}' + (-1)^{V_{3k+2}'} G_b(V_{3k-2},V_{3k-1}',V_{3k},V_{3k-1})]$$

$$[V_{3k-3} + (-1)^{V_{3k+3}} G_b(V_{3k-1}',V_{3k},V_{3k-1},V_{3k-2}')]$$

$$[V_{3k+4} + (-1)^{V_{3k+4}} G_b(V_{3k},V_{3k-1},V_{3k-2}',V_{3k+3})] \Big\} \ .$$

$$\prod_{\ell=1}^{r} [V_{j+1-\ell}' + (-1)^{V_{j+1-\ell}'} G_b(V_{j-\ell-3},V_{j-\ell-2}',V_{j-\ell-1},V_{j-\ell})]$$

where r = MOD(j-1,3). It is very important to note that the variables in the product expression match successively. That is, V_{3k-1} and V_{3k+2} all have the prime notation in each sub-expression. At this point, there are three possible cases:

Case 1, r = 0

In this case, the proof is completed by a series of change of variables as summing over V_i is the same as summing over V_i'. The remainder of the proof becomes

$$=\sum P_b(V_1,V_2',V_3,V_4)\cdot \prod_{k=1}^{j} [V_k+(-1)^{V_k} G_b(V_{k-4},V_{k-3},V_{k-2},V_{k-1})] \tag{A.11}$$

$= P_b(V_1,V_j)$.

Case 2, r = 1

In this case, the remainder of the proof becomes

$$\begin{aligned} &= \quad P_b(V_1,V_2',V_3,V_4) \prod_{k=1}^{j-1} [V_k'+(-1)^{V_k'} G_b(V_{k-4},V_{k-3}',V_{k-2},V_{k-1})] \cdot \\ &\qquad \cdot [V_j'+(-1)^{V_j'} G_b(V_{j-4},V_{j-3}',V_{j-2},V_{j-1})] \\ &= \quad P_b(V_1,V_2',V_3,V_4) \prod_{k=1}^{j-1} [V_k+(-1)^{V_k} G_b(V_{k-4},V_{k-3}',V_{k-2},V_{k-1})] \\ &\qquad \cdot [V_j'+(-1)^{V_j'} G_b(V_{j-4},V_{j-3},V_{j-2},V_{j-1})] \ . \end{aligned} \tag{A.12}$$

Thus, in this case $P_a(V_1,V_j) = P_b(V_1,V_j')$. This implies that $P_a(0_1,0_j) = P_b(0_1,1_j)$; $P_a(0_1,1_j) = P_b(0_1,0_j)$; $P_a(1_1,0_j) = P_b(1_1,1_j)$; $P_a(1_1,1_j) = P_b(1_1,0_j)$. We know $P_a(0_1,1_j) = P_a(1_1,0_j)$ and $P_b(0_1,1_j) = P_b(1_1,0_j)$ and $P(0) = P(1) = 0.5$. So $P_a(V_1,V_j) = P_b(V_1,V_j) = 0.25$ for all V_1 and V_j.

Case 3, $r = 2$

In this case, as in the case where $r = 0$, the remainder of the proof is straightforward, requiring only a change of variables.

APPENDIX B

GOODNESS-OF-FIT TEST FOR N-GRAMS

We may think of the one-dimensional binary texture generation process as an experiment which should yield certain N-grams given certain generation parameters. These N-grams or Nth-order densities may be determined analytically by combining Eq. (2.14) and Eq. (2.24) of Chapter 2. However, as with any Monte-Carlo simulation, the analytic parameters may not agree exactly with the statistics or estimates of those parameters based on observations from the simulation or experiment. Statistical tests may be used to determine whether the statistics match the parameters to the extent expected given a given random sample.

The most common statistical test for this purpose is the "goodness-of-fit" test involving the chi-square distribution. This test is used when we want to compare an observed distribution with the corresponding values of a theoretical distribution. The test is based on the statistic

$$\chi^2 = \sum_{i=1}^{2^N} \frac{(f_i - e_i)^2}{e_i} \quad . \tag{B.1}$$

The f_i and the e_i are the observed and expected frequencies of a certain N-gram pattern in our experiment. The sampling distribution of this statistic is approximately the chi-square distribution with 2^{N-1} degrees of freedom. (The constraints on the system of N-grams yield this number of degrees of freedom.) A chi-square distribution with n degrees of freedom is given by

$$f(\chi^2) = \frac{(\chi^2)^{\frac{1}{2}(n-2)}}{2^{n/2}\Gamma(\frac{n}{2})} \ell^{-\chi^2/2} \tag{B.2}$$

The approximation is close provided e >5.

As a first step in the testing process, the complete set of N-grams, $P(V_1, V_2, \ldots, V_N)$ is computed using Eq. (2.14) and Eq. (2.24). Their corresponding statistics, $\hat{P}(V_1, V_2, \ldots, V_N)$ are computed from the generated texture by counting the number of occurrences of each pattern $(V_1, \ldots, V_N)$ and dividing by the total sample size, M. Then, $f_i = \hat{P}(V_1, \ldots, V_N) \cdot M$ and $e_i = P(V_1, \ldots, V_N) \cdot M$. For large N, the number of degrees of freedom is greater than 30 and a normal distribution approximation is quite accurate. The expression $\sqrt{2\chi^2} - \sqrt{2N-1}$ is approximately normally distributed as the standard normal distribution. The tables listed in [18] may also be used for large N. Additional details concerning the goodness-of-fit test may be found in [45,63-65].

For each of the textures in Chapter 2, this test was applied to each texture half and the hypothesis that the expected and measured N-grams are statistically equal was accepted at a $\alpha = 0.01$ confidence level for $N = 1,\ldots,10$. Actually, these tests are not independent in a statistical sense as the information content for each N overlaps. But this poses no violation of assumptions of the statistical test. However, all possible patterns in the generated sample may be a violation of the independence of samples assumption. Still, a low chi-square value indicates a good fit of the data to predicted parameter.

APPENDIX C

PSEUDO-RANDOM VARIATE GENERATION

Two algorithms were used in this thesis to generate uniformly-distributed pseudo-random numbers. Both are modifications of the multiplicative congruence method [48,66]. These uniformly-distributed variates may be transformed to normally-distributed deviates using the inverse normal probability distribution function.

The first algorithm is the Lehner Pseudo-Random Number Generator [67]. The general form of this generator is

$$X_{n+1} = KX_n (\text{MOD } 2^{31}-1) \qquad \text{(C.1)}$$

where

$$K = 14^{29} (\text{MOD } 2^{31}-1) = 630360016_{10}$$

The resulting number is basically a 31-bit pattern which is in the form of a floating-point number in the range (0,1). The algorithm can be written to avoid division by $2^{31}-1$ [68].

The second algorithm used to generate uniformly distributed variates is

$$X_{n+1} = 7^5 X_n (\mathrm{MOD}\ 2^{31}-1) \quad . \tag{C.2}$$

The resulting integer is multiplied by 2^{-31} to convert it to a floating-point number. This generator is reported in Refs. [69-71].

These uniform deviates may be transformed to normally-distributed deviates if desired. this is accomplished by computing the inverse of the integral

$$X = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{Y} e^{-\ t^2/2}\ dt \quad . \tag{C.3}$$

The algorithm for accomplishing this is described in Ref. [57] and is briefly outlined here.

The basic interval (0,1) is divided into 4 segments. In each segment the inverse of the Gaussian integral, invgauss(X), or an integral of a similar form is approximated by a minimax rational function [55,72]. The approximations for the final 3 segments (comprising the interval $X \varepsilon \{(0,0.075)\ (0.925,1)\}$) are functions of the transformed variable

$$W = \sqrt{-\log e(1-X^2)} \quad . \tag{C.4}$$

This transformation of the variable improves the efficiency and stability of the approximation [60]. The rational functions in these intervals are of the form

$$\text{invgauss}(X) \cong W+W\cdot\left(C_0+\frac{C_1W+C_2W^2+C_3W^3}{d_0+d_1W+d_2W^3+W^3}\right) . \tag{C.5}$$

For the remainder of the (0,1) interval, $x\,\varepsilon\{(0,075,\ 0.925)\}$, the function is of the form

$$\text{invgauss}(X) = \\ (Z+Z\cdot(b_0+a_1\cdot Z^2/(b_1+Z^2+a_2/(b_2+Z^2+a_3/(b_3+Z^2)))))\ *\text{SGN}(Z) \tag{C.6}$$

where $z = |1-2x|$. The constants a_i, b_i, c_i and d_i have specific values found by solving the minimax problem and are given in Ref. [57].

REFERENCES

[1] P. Brodatz, Textures: A Photographic Album for Artists and Designers. New York: Dover, 1966.

[2] B.H. McCormick and S.N. Jayaramamurthy, "Time Series Models for Texture Synthesis," Int. J. of Computer and Info. Sciences, vol. 3, pp. 329-343, Dec. 1974.

[3] S.R. Purks and W. Richards, "Visual Texture Discrimination Using Random Dot Patterns," J. Opt. Soc. Am., vol. 67, pp. 765-771, June 1977.

[4] A. Gagalowicz, "Stochastic Texture Fields Synthesis From A Priori Given Second Order Statistics," Proc. Conf. on Pattern Recognition and Image Processing, pp. 376-381, Chicago, August 6-8, 1979.

[5] A. Gagalowicz, "Visual Discrimination of Stochastic Texture Fields Based Upon Their Second Order Statistics," Proc. of Fifth Int. Joint Conf. on Pattern Recognition, Miami Beach, Dec. 1980, pp. 786-788.

[6] K. Abend, T.J. Harley, L.N. Kanal, "Classification

of Binary Random patterns," IEEE Trans. on Info. Theory, vol. IT-11, no. 4, pp. 538-544, Oct. 1965.

[7] R.W. Conners, Some Theory on Statistical Models for Texture and Its Application to Radiographic Image Processing, Ph.D. Thesis, College of Engineering, Univ. of Missouri, Columbia, Dec. 1976.

[8] B. Julesz, "Visual Pattern Discrimination," IRE Trans. on Info. Theory, vol. IT-8, pp. 84-92, Feb. 1962.

[9] D.D. Garber, "Models for Texture Analysis and Synthesis," USCIPI REport 910, Image Processing Institute, Univ. of Southern California, Los Angeles, Sept. 1979.

[10] S.Y. Lu and K.S. Fu, "Stochastic Tree Grammar Inference for Texture Synthesis and Discrimination," Proc. Conf. on Pattern Recognition and Image Processing, Chicago, May 1978, pp. 340-345.

[11] J.T. Tou, D.B. Kao, and Y.S. Chang, "Pictorial Texture Analysis and Synthesis," Proc. of Third Int. Joint Conf. on Pattern Recognition, Coronado, Calif., Nov. 1976, p. 590.

[12] K. Deguchi and I. Morishita, "Texture

Characterization and Texture-Based Image Partitioning Using Two-Dimensional Linear Estimation Techniques," IEEE Trans. on Computers, vol. C-27, pp. 739-745, Aug. 1978.

[13] H. Kaiser, A Quantification of Textures on Aerial Photographs. Note 121, AD 69484, Boston Univ. Research Labs., Boston, 1955.

[14] W.K. Pratt, O.D. Faugeras, and A. Gagalowicz, "Visual Discrimination of Stochastic Texture Fields," IEEE Trans. on Syst., Man and Cybern., vol. SMC-8, no. 11, pp. 794-804, Nov. 1978.

[15] R.M. Haralick, K. Shanmugam, and I. Dinstein, "Textural Features for Image Classification," IEEE Trans. on Syst., Man and Cybern., vol. SMC-3, pp. 610-621, Nov. 1973.

[16] I. Pollack, "Discrimination of Third-Order Markov Constraints within Visual Displays," Perception and Psychophysics, vol. 13, no. 2, pp. 276-280, 1973.

[17] D. Maki and M. Thompson, Mathematical Models and Applications. Englewood Cliffs, New Jersey: Prentice-Hall, 1973.

[18] W.H. Beyer, Handbook of Tables for Probability and

Statistics, Cleveland: The Chemical Rubber Company, 1968.

[19] D.D. Garber, "One-Dimensional Texture Pattern Generation and Discrimination," USCIPI Report #840, Image Processing Inst., Univ. of Southern California, Sept. 1978.

[20] N. Draper and H. Smith, Applied Regression Analysis, New York: John Wiley & Sons, 1966.

[21] D.F. Morrison, Multivariate Statistical Methods, New York: McGraw-Hill, 1967.

[22] G.W. Stewart, Introduction to Matrix Computations, New York: Academic Press, 1973.

[23] F.D. Russell, Predetection and Postdetection Filtering for Improved Resolution in Optical Systems, Technical Memo 70-007, Philco-Ford Corporation, Palo Alto, California, Aug. 1970.

[24] B. Julesz, E. Gilbert, and J.D. Victor, "Visual Discrimination of Textures with Identical Third order Statistics," Biological Cybernetics, vol. 31, pp. 137-140, 1978.

[25] G.E.P. Box and G.M. Jenkins, Time Series Analysis: Forecasting and Control, San Francisco: Holden-Day,

1976.

[26] O.D. Faugeras and W.K. Pratt, "Decorrelation Methods of Texture Feature Extraction," IEEE Trans. on Patt. Recog. and Machine Intelligence, vol. PAMI-2, no. 4, pp. 313-323, July 1980.

[27] R. Gordon, R. Bender, and G.T. Herman, "Algebraic Reconstruction Techniques (ART) for Three-Dimensional Electron Microscopy and X-ray Photography," J. Theor. Biol., vol. 29, pp. 471-481, 1970.

[28] R. Gordon, "A Tutorial on ART," IEEE Trans. on Nuclear Sciences, vol. NS-21, pp. 78-93, June 1974.

[29] O.D. Faugeras, "Autoregressive Modeling with Conditional Expectations for Texture Synthesis," Proc. of Fifth Int. Joint Conf. on Pattern Recognition, Miami Beach, Dec. 1980, pp. 786-788.

[30] D.R. Cox and P.A.W. Lewis, The Statistical Analysis of Series of Events, London: Chapman and Hall, 1966.

[31] E.J. Hannan, Time Series Analysis, London: Chapman and Hall, 1960.

[32] C.W. Ostrom, Time Series Analysis, Beverly Hills:

Sage Publications, 1978.

[33] M. Nerlove, D. Brether, and J. Carvalho, Analysis of Economics Time Series, New York: Academic Press, 1979.

[34] S.C. Hiullmer and G. Tiao, "Likelihood Function of Stationary Multiple Autoregressive Moving Average Models," Journal of the American Statistical Association, vol. 74, no. 367, pp. 652-661, Sept. 1979.

[35] A.F. Siegel, "Testing for Periodicity in Time Series," Journal of the American Statistical Association, vol. 75, no. 370, pp. 345-349, June 1980.

[36] D.R. Cox, The Analysis of Binary Data, London: Chapman and Hall, 1970.

[37] B. Kedem, "Estimation of the Parameters in Stationary Autoregressive Processes After Hard Limiting," Journal of the American Statistical Association, vol. 75, no. 369, pp. 146-154, March 1980.

[38] B. Bowerman and R.O'Connell, "An Intuitive, Systematic Approach to Tentatively Identifying a

Box-Jenkins Seasonal Model," _1979 Proceedings of the Business and Economic Statistics Section of the American Statistical Association_, Washington, D.C., Aug. 13-16, 1979, pp. 330-342.

[39] F.A. Graybill, _Theory and Application of the Linear Model_, North Scituate, Massachusetts: Duxbury Press, 1976.

[40] B.R. Hunt, "The Application of Constrained Least Squares Estimation to Image Restoration by Digital Computer," _IEEE Trans. on Computers_, vol. C-22, no. 9, pp. 805-812, Sept. 1973.

[41] H.C. Andrews, _Digital Image Restoration_, New Jersey: Prentice-Hall, 1977.

[42] W.K. Pratt, _Digital Image Processing_, New York: John Wiley & Sons, 1978.

[43] A.E. Hoerl and R.W. Kennard, "Ridge Regression: Biased Estimation for Non-Orthogonal Problems and Applications," _Technometrics_, vol. 12, no. 55, 1969.

[44] J.R. Alldredge and N.S. Gilb, "Ridge Regression: An Annotated Bibliography," _Int. Statist. Rev._, vol. 44, no. 355, 1976.

[45] M. Kendall and A. Stuart, The Advanced Theory of Statistics, Volume 1, New York: MacMillan, 1977.

[46] M. Kendall and A. Stuart, The Advanced Theory of Statistics, Volume 2, New York: MacMillan, 1979.

[47] M. Kendall and A. Stuart, The Advanced Theory of Statistics, Volume 3, London: Griffin, 1976.

[48] D.E. Knuth, The Art of Computer Programming, Volume 2, Menlo Park, California: Addison Wesley, 1969.

[49] R.M. Haralick, "Statistical and Structural Approaches to Texture," Proc. of the IEEE, vol. 67, no. 5, May 1979.

[50] R.M. Haralick and R. Bosley, "Texture Features for Image Classification," Third ERTS Symp., NASA SP-351, NASA Goddard Space Flight Center, Greenbelt, MD, pp. 1929-1969, Dec. 10-15, 1873.

[51] J.S. Weszka, C.R. Dyer, and A. Rosenfeld, "A Comparative Study of Texture Measures for Terrain Classification," IEEE Trans. on Syst., Man and Cybern., vol. SMC-6, pp. 269-285, April 1976.

[52] P. Chen and T. Pavlidis, Segmentation by Texture Using a Coocurrence Matrix and a Split-and-Merge

Algorithm. TR-237, Princeton Univ., Princeton, NJ, Jan. 1978.

[53] H. Kaizer, A Quantification of Textures on Aerial Photographs, Note 121, AD 69484, Boston Univ. Research Labs., Boston, 1955.

[54] C.Y. Kramer, A First Course in Methods of Multivariate Analysis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1972.

[55] J.F. Hart et al., Computer Approximations, New York: Wiley, 1968.

[56] B.R. Hunt and T.M. Cannon, "Nonstationary Assumptions for Gaussian Models of Images," IEEE Trans. on Syst., Man and Cybern., pp. 876-881, Dec. 1976.

[57] P. Kinnucan and H. Kuki, A Single Precision Inverse Error Function Subroutine, Computation Center, The University of Chicago, Chicago, Illinois.

[58] T.H. Naylor et al., Computer Simulation Techniques, New York: John Wiley & Sons, Inc., 1966.

[59] F.D. Russell, "Non-Redundant Arrays and Post Detection Processing for Aberration Compensation in

Incoherent Imaging," J. Opt. Soc. Am., vol. 61, no. 2, Feb. 1971.

[60] A.J. Strecok, "On the Calculation of the Inverse of the Error Function," Mathematics of Computation, vol. 22, no. 101, pp. 144-158, 1968.

[61] J. Leech, "On the Representation of 1,2,...,n by Differences," Journal of the London Mathematical Society, vol. 31, part I, pp. 160-169, April 1976.

[62] A.J. Lawrence, "Partial and Multiple Correlation for Time Series," The American Statistics, vol. 33, no. 3, Aug. 1979.

[63] J.E. Freund, Mathematical Statistics, New Jersey: Prentice-hall, 1971.

[64] H.L. Alder and E.B. Roessler, Introduction to Probability and Statistics, San Francisco: W.H. Freeman and Company, 1972.

[65] I. Miller and J.E. Freund, Probability and Statistics for Engineers, New Jersey: Prentice-Hall, Inc., 1965.

[66] G. Struble, Assembler Language Programming: The IBM System/360, Menlo Park, Calif.: Addison-Wesley, 1971.

[67] W.H. Payne, J.R. Rabring, and T.P. Bogyo, "Coding the Lehmer Pseudo Random Number Generator," Communications of the ACM, pp. 85-86, Feb. 1969.

[68] PDP-10 Science Library and FORTRAN Utility Subprograms, DEC-10-SFLE-D, Digital Equipment Corporation, Maynard, Massachusetts.

[69] P.A. Lewis and A.S. Goodman, "Pseudo-Random Number Generator for the System/360," IBM Systems Journal, no. 2, 1969.

[70] G.P. Learmonth and P.A.W. Lewis, Naval Postgraduate School Random Number Generator Package LLRANDOM, NPS55LW 73061A, Naval Postgraduate School, Monterey, California, June 1973.

[71] G.M. Phillips and P.J. Taylor, Theory and Applications of Numerical Analysis, New York: Academic Press, 1973.

[72] D.D. Garber and A.A.Sawchuk, "Computational Models for Texture Analysis and Synthesis," Proc. of SPIE's Technical Symposium East 1981, Vol. 128, April 1981.

[73] D.D.Garber and A.A.Sawchuk, "Texture Simulation Using a Best-fit Model", Proc. Conf. on Pattern

Recognition and Image Processing, Dallas, August 1981.

[74] O.D.Faugeras and D.D.Garber, "Algebraic Reconstruction Techniques for Texture Synthesis," Proc. of Fifth Int. Joint Conf. on Pattern Recognition, Miami Beach, December 1980, pp. 783-785.